2023

A Multinomial Classification And Prediction Model Utilizing Deep Learning For Malware Detection On Raspberry Pi Internet Of Things Devices Using Unrestrained Network Connections

Thomas A. Woolman
Indiana State University

A MULTINOMIAL CLASSIFICATION AND PREDICTION MODEL UTILIZING DEEP LEARNING FOR MALWARE DETECTION ON RASPBERRY PI INTERNET OF THINGS DEVICES USING UNRESTRAINED NETWORK CONNECTIONS

A dissertation

Presented to

The College of Graduate and Professional Studies

Bailey College of Engineering and Technology

Indiana State University

Terre Haute, Indiana

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

Thomas A. Woolman

May 2023

Keywords: deep learning, neural network, Internet of Things, malware, hypothesis testing

VITA

Thomas A. Woolman

EDUCATION

2023	Indiana State University, Terre Haute, Indiana Ph.D. in Technology Management, Digital Communications Systems concentration
2016	Saint Joseph's University, Philadelphia, Pennsylvania Master of Business Administration, Data Analytics concentration
2015	Emporia State University, Emporia, Kansas Master of Science, Earth Science concentration
2015	Emporia State University, Emporia, Kansas Graduate Certificate, Geospatial Analytics
2013	Saint Joseph's University, Philadelphia, Pennsylvania Master of Science, Data Analytics
1995	Drexel University, Philadelphia, Pennsylvania Bachelor of Science, Information Systems

PROFESSIONAL EXPERIENCE

1998-Present	On Target Technologies, Inc., Ridgeley, West Virginia Principal Data Scientist
2022-Present	Fountain Global, LLC, Fairfax, Virginia Chief Data Scientist
2017-2022	Southern New Hampshire University, Manchester, New Hampshire Adjunct Professor, College of Continuing and Professional Education
2015-Present	SurgeonCheck, LLC, Bethlehem, Pennsylvania Director of Data Science
2006-2014	Tanzanian Gold Corporation, Dar es Salaam, Tanzania Managing Consultant, Biogeochemistry Exploration Department

COMMITTEE MEMBERS

Committee Chair: John Pickard, PhD

Associate Professor, Department of Technology Systems

College of Engineering and Technology

East Carolina University

Committee Member: Philip Lunsford, PhD

Associate Professor Emeritus, Department of Technology Systems

College of Engineering and Technology

East Carolina University

Committee Member: Carroll M. Graham, EdD

Professor, Department of Human Resource Development and Performance Technologies

Bailey College of Engineering and Technology

Indiana State University

ABSTRACT

This study explored the use of deep learning artificial intelligence, machine learning, and non-parametric statistics for successfully detecting various classes of malware attacks against Raspberry Pi hardware devices over unrestrained digital communication networks. Furthermore, the use of permutated tests of statistical significance were applied to these artificial intelligence and machine learning models for the purpose of providing scientific evidence for the statistical significance of the findings from the predictive classification models.

Much effort has taken place in recent years to apply various types of machine learning (non-parametric statistical learning systems) to use case problems involving network intrusion detection and malware identification. This research provides thoroughly tested methodologies that provide a wealth of prediction capabilities for this problem space, with little if any attempt at scientific proof of the effectiveness of these predictions. Much was being predicted in the field of applied machine learning and artificial intelligence for digital communication systems, but little was being scientifically proven.

In general, the application of scientific tests of falsifiability have been lacking in this field because of the gap that exists between the non-linear, non-parametric nature of machine learning models and the generally accepted methodologies for providing scientific evidence of causal relationships. Scientific tests of falsifiable hypothesis testing are rooted in the general linear model, such as linear regression, logistic regression, ANOVA, MANOVA, etc. Because those classical statistical methods address linear relationships between the independent and dependent

variables, they are generally incapable of adequately analyzing relationships and predictions developed by these non-parametric statistical learning system models.

Literature reviews support the principal concept for this research that non-parametric statistical learning systems can be the subject of statistical hypothesis testing based on classical general linear model-based statistical methods, under specific frameworks with which to create regularization effects. Thus, by incorporating permuted tests of statistical significance using methods such as PERMANOVA, scientific tests of statistical significance were able to take place over the course of this research.

The net result was the development of a defense in depth study across a range of different types of non-parametric statistical learning models that analyzed various aspects and use cases of malware across Raspberry Pi networked devices. Various types of permuted tests of statistical significance took place to produce non-parametric p-values as well as measures of non-Euclidean distance tests of homogeneity between groups, to produce non-parametric visualization outputs. These non-parametric validation models successfully analyzed various cause and effect relationships responsive to the four separate research questions, addressed through null and alternative hypothesis statements. The permuted tests of statistical significance and non-parametric measurement tests of homogeneity analyzed the predicted output from the cohort of supervised learning and unsupervised learning predictive models, providing strong scientific evidence for the effectiveness of artificial intelligence models applied to this specific domain.

PREFACE

This dissertation is original, unpublished, independent work by the author, Thomas A. Woolman. The datasets utilized in this research that were analyzed by the artificial intelligence and machine learning models developed by the author was freely made available to me from the Aposemat IoT-23 research.

The author utilized the Aposemat IoT-23 data for the ex post facto research in this study, available as a freely distributed data product created at the Avast-AIC laboratory supported with the funding of Avast Software. It was freely available for use in academic and commercial research with attribution, and the author thanks this laboratory for the use of their valuable data.

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude and deepest appreciation for all who have supported and encouraged me to complete this dissertation. I also sincerely thank all of my professors who have been a part of my doctoral program at Indiana State University, including my professors in the Technology Management PhD consortium program at East Carolina University, Bowling Green State University, University of Central Missouri, and North Carolina A&T State University.

Dr. Philip J. Lunsford II, thank you for your support and for the opportunity to conduct joint research with you that resulted in our publication in an international peer reviewed academic journal.

Thanks to Dr. Carroll M. Graham for his support and for providing valuable guidance regarding aspects of technology management for human resource development for this dissertation.

Dr. John L. Pickard, thank you for advising me as my dissertation committee chair, providing valuable guidance and support during this dissertation project. Thank you as well for conducting joint research with me outside of this project, encouraging me to create additional research publications based on some of the methodologies utilized in this work.

Lastly, I would like to thank my wife Brittnay L. Woolman, R.N. and my children Emily and Joshua. Their tireless patience, support and encouragement are what made not only this dissertation possible but also enabled my entire graduate school career. Since they were born, my

children (13-year-old twins as of this publication date) have not known a time when their Dad was not in one or more graduate school programs. The conclusion of this dissertation work finalizes my graduate school career and I have my family to thank for making this all possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Chapter Overview

This research used deep learning and machine learning algorithms to investigate issues related to the detection of malware and network intrusion attacks against Raspberry Pi devices on unrestrained network connections, for the purpose of applying a novel application framework to conduct parametric, asymptotic hypothesis testing to nonparametric, non-asymptotically converged statistical learning models. This allowed for the application of scientific falsifiability testing to the findings of these nonparametric models as well as establishing causal relationships in the population. This research is potentially critical, as it addresses cyber defense shortcomings that impact a wide range of increasingly ubiquitous hardware devices that comprise the Internet of Things (IoT), including a growing use in some networked medical devices (Sengan et al., 2022).

However, there is an overall lack of research regarding the application of the scientific method in terms of the use of falsifiability hypothesis testing with which to determine the statistical significance for the effective application of deep learning and machine learning algorithms for this use case. This is due to a fundamental conflict between the nature of classic and inflexible linear regression models with which to conduct statistical hypothesis testing and nonparametric deep learning models. Because nonparametric statistical learning models such as

deep learning, multi-layer perceptron neural networks primarily utilize nonlinear representations within their hidden layers, there is to date little to no research that closes the gap between the interpretability of traditional statistical models and the flexibility of machine learning and deep learning systems, in the field of digital communication systems.

The overall objective of this research is to provide a series of hybrid solutions that integrate the respective capabilities of deep learning systems and classical statistics, with the interpretability and causal inference ability of asymptotic general linear regression models.

Technology Management Impacts

This research integrates with the larger field of Technology Management by addressing an ongoing need in securing enterprise resources and maintaining competitive advantages. Enterprises are continuously subject to network attacks in the form of evolving malware threats directed at IoT network data. Digital communication network managers and administrators experience this growing problem in both industry and government. This research endeavors to prove with scientific efficacy that aspects of artificial intelligence systems can supply an effective defense in depth strategy to mitigate this management challenge.

By addressing this ongoing threat to ever-increasing amounts of IoT data on homogenous classes of hardware over unrestrained networks utilizing aspects of artificial intelligence and machine learning, the research also addresses the human factor present in technology management. The research addresses complex features and patterns discovered by the AI/ML models in making the various classification predictions and employs a secondary series of AI/ML models to produce explainable characteristics about individual class predictions.

These explainable features for specific class predictions address aspects of human resource development needs for human network administrators and technology managers within

the enterprise, thus leveraging the corporate AI/ML models as a mechanism to increase our understanding of the digital communication network threats.

Background

In this chapter, the author provides an overview of the study by first providing a general background as well as context to the research problem as well as the research aims, questions, and objectives along with the significance of the research and, finally, the limitations. Likewise, this provides the summary information for this topic as it relates to the fitness and merit of the research focus as well as its potential for scientific contribution to the field of study. In the study, the author used deep learning and machine learning algorithms to model nonparametric network anomalies to predict as well as detect indications of the presence of malware on unrestrained network connections that are acting upon Internet of Things (IoT) devices that are Raspberry Pi-compliant hardware devices.

The problem for this study involves the use of nonparametric statistical learning methods, including machine learning and deep learning algorithms, to develop models for the detection of a wide range of known, distinct malware entities upon unrestrained network connections for Internet of Things (IoT) devices utilizing the Raspberry Pi family of single-board computer specifications (IoT-enabled industrial Raspberry Pi compatible devices). The dataset (IoT-23) used for this study incorporated human-labeled curated analysis of individual network traffic flows from IoT devices in controlled environments, consisting of 20 malware captures and three captures for benign IoT device traffic. The laboratory first published this dataset in January 2020, with captures ranging from 2018 spanning through 2019.

First, the researcher describes the IoT network flow dataset, providing detailed analysis and summary statistics of the flow metadata. Next, a review of the history and methodology of

the capture processes used to obtain the IoT-23 dataset utilizing the Raspberry Pi family of single-board computer specifications (IoT-enabled industrial Raspberry Pi compatible devices is provided. The author reviews the research methodologies that consists of the problem statement, the research questions, and the hypotheses. A justification for the research need, as well as the assumptions and limitations underlying the proposed methodology is given. The author identifies major assumptions and limitations along with a review of key terms, discussed in context with the research problem.

The objectives of this research are to use the findings to:

Determine the effectiveness of an ensemble deep learning algorithm and statistical analysis framework at performing multinomial classification of IoT malware anomalies, utilizing network flow data.

Identify the impacts of the various independent variables within the unrestrained IoT Internet traffic flows on the class detection accuracy for the various IoT malware threat anomalies.

Quantify some of the benefits realized from the use of a deep neural network for supervised learning for this use case, including determining the effect of random chance through inter-rater reliability for the multinomial classification.

Develop a model that can be used to classify with probability scoring in near real-time for a malware threat determination when an IoT device on an unrestrained Internet connection is subjected to various malware anomaly threats.

Apply aspects of scientific falsifiability to the predictive classification ensemble model for rejecting or failing to reject the null hypothesis. The purpose of this objective is to

demonstrate that aspects of statistical significance testing are potentially applicable using novel methods applied to AI/ML models that have significant nonlinear sensitivities.

Statement of the Problem

As the Internet of Things vastly increased in size to 14.4 billion active connections in 2022 (IoT.Business.News, 2022) representing an 18% annualized increase that is projected to reach approximately 27 billion active connections by 2025, a growing imperative exists to help secure this increasingly critical aspect of global digital infrastructure. Traditional rules-based and heuristic-based cyber defensive systems are often inadequate to address the unique challenges, risks, and limitations inherent in IoT communications architectures. Consequently, the need for sophisticated, adaptable defensive cyber systems as part of a defense in depth strategy that is scientifically proven to be demonstrably effective across a broad range of risk categories utilizing falsifiable hypothesis testing has never been greater.

Research Questions and Hypothesis

In this study, the researcher used hybrid linear and nonlinear statistical analysis to provide a framework to ascertain the statistical significance. Those statistically significant findings of the nonparametric machine learning and deep learning models investigate the ability of the AI/ML models to predict the presence of distinct classes of malware. Specifically, within an Internet of Things (IoT) unrestrained network traffic flow dataset in devices utilizing the Raspberry Pi family of single-board computer specifications (IoT-enabled industrial Raspberry Pi compatible devices). This development of research using innovative and traditional statistical hypothesis testing provided a measure of falsifiability for the machine learning and deep learning predictive models. This is conceptually novel to the field of deep learning and machine learning for digital communication network analytics. Such scientific findings have been a challenge in

the past, due to the highly nonparametric nature of machine learning and deep learning prediction models that often do not consistently produce simple linear relationships between independent and dependent variables to provide valid inferences.

Since the purpose of this research is to propose an ensemble series of nonparametric statistical learning models to address various research questions that are addressed through novel adaptations of asymptotic hypothesis testing, the research addressed how these process models can be used to accurately identify and provide falsifiable predictive models. These models addressed research questions related to the inferences that lead to the detected presence of IoT malware and network intrusions instances that are known to or potentially might be actively attacking IoT devices of various architectures through an unrestricted Internet connection. The author utilized aspects of scientific reproducibility and falsifiability measures to statistically assess findings using a novel use of statistical significance testing that is applied to non-linear machine learning and deep learning models. The substantive research questions, therefore, are as follows:

RQ1: What do the network flow variables reveal about the distribution of malware classifications in the IoT-23 dataset and the ability to construct non-parametric predictions for the known malware classes?

RQ2: What are the distinct independent variable cohort relationships between the network traffic flow variables and specific individual IoT malware classes when conducting nominal Internet communications activities upon unrestrained networks?

RQ3: What effect does a supervised nonparametric statistical learning model have on determining the presence of malware on human-curated datasets for Raspberry Pi devices using network flow data across an unrestrained network?

RQ4: What effect does an unsupervised, self-supervised predictive model have for predicting malware presence without utilizing human-curated labeled training data on Raspberry Pi devices using network flow data?

The author augmented the exploration of supervised machine learning and deep learning models in the first three research questions by various non-parametric models to support a multi-layer defense in-depth strategy driven by machine learning and artificial intelligence models. The research incorporated demonstrated falsifiability using the model findings to provide permuted statistical scientific evidence for the effectiveness and applicability of these statistical learning models.

The first research question is primarily exploratory, to allow for the investigation of the association between the network traffic flow attributes and the various labeled individual malware classes. The research examined the parametric and nonparametric relationships of the independent variables in the IoT-23 network flows to the dependent variable (malware class) of the dataset to understand their ability to contribute to a nonparametric statistical learning predictive model. This process represents the linear and nonlinear statistical significance of each independent variable in terms of the relative effect on the model, transformed and scaled to show relative importance on a percentile basis.

As such, it is effectively a proof of a precursor or “pathfinder” method for determining the suitable presence of a non-parametric statistical signal being present in the dataset between one or more independent variables and the curated dependent variable in the network flow data. This pathfinder method can potentially demonstrate the suitability for the employment of AI/ML predictive classification models that can produce a more refined non-parametric statistical

classification model, including the determination of the optimal key response indicator independent variables in the model.

This first research question required construction of a non-parametric multivariate analysis of variance (NPMANOVA), also known as a permuted multivariate analysis of variance (PERMANOVA) model. Specifically, the researcher employed a one-way NPMANOVA to address the research question, as a one-way MPMANOVA is the non-linear (non-parametric) extension of the one-way multivariate analysis of variance (one-way MANOVA). Like the MANOVA, the NPMANOVA is primarily used to determine if statistically significant differences exist between two or more independent groups (in this case, our labeled malware class groups represented by a categorical (nominal type) independent variable, in relation to two or more continuous dependent variables. The one-way MANOVA is an extension of the one-way ANOVA, used when analyzing only one dependent variable at a time.

NPMANOVA is thus a non-parametric multivariate test used to compare the multivariate distribution of multiple independent groups and is useful for adapting to machine learning prediction models, which are primarily asymptotic prediction methods. This research used the NPMANOVA model for this first research question to test the following hypotheses:

$H1_0$: The centroids of each of the distinct malware groups' multivariate dependent network flow variables in the IoT-23 dataset are equal.

$H1_a$: There is at least one pair of malware groups with significantly unequal multivariate dependent variable centroids in the network flow dataset.

Research question two involved a more detailed exploration of the non-parametric classification predictive model relationships of the multiple independent variables as they relate to each specific class label prediction (dependent variable) by the primary predictive

classification machine learning/artificial intelligence model. This exploratory analysis utilized SHAP (SHapley Additive exPlanations) calculations to provide in-depth explanations for each class label prediction. A confirmation of the statistical significance of the SHAP model explanation output was made with a gradient-boosted machine learning algorithm-based model, used to create a baseline multinomial classification prediction. The author then analyzed this classification model to discover Shapley values based on calculated coalition relationships (between IV coalitions), for each of the marginal contributions of *F*, and then finding the mean *F* marginal contribution for each coalition across the dataset. This set of Shapley values, where *K* is the sum of all features in the model represent a metadata subset of 2^K potential coalitions, with the Shapley values scaling exponentially with the number of features (*F*) contained in the dataset.

The researcher conducted a hypothesis test for this research question by first using a k-fold cross-validation testing of SHAP (SHapley Additive exPlanations) calculations produced by a non-parametric gradient-boosted machine (GBM) algorithm. This method employed asymptotic normal predictions using subsampled prediction values to approximate normal predictions for exploitation using hypothesis tests and confidence intervals analogous to a parametric framework. The use of a permuted multiple linear regression model established that a statistically significant relationship exists between the continuous type data contained within the Shapley coalition contribution score permutations and the categorical type dependent variable containing the various labeled malware classes.

The hypothesis to address the second research question is:

$H2_0$: An independent variable as it contributes to the asymptotic prediction for the dependent variable of malware class (nominal categorical factor variable) is not important, i.e. $\psi_{0,0,j} = 0$ for some j.

$H2_a$: An independent variable as it contributes to the asymptotic prediction for the dependent variable of malware class (nominal categorical factor variable) is important, i.e. $\psi_{0,0,j} > 0$ for some j.

The author addressed the third research question through an analysis of the nonparametric statistical learning methods utilized in RQ1 and RQ2, utilizing subsampled prediction output averaging to produce asymptotically normalized predictions. By estimating the variance of these predictions, the research exploited model output properties that produced normalized hypothesis tests and confidence intervals analogous to linear statistical tests produced within traditional, parametric frameworks. Next, a deep learning multinomial (multiple) classification predictive model to create an asymptotic model based on subsampled gradient boosted machine predicted model non-parametric output. The deep learning predictive model output was addressed with a permuted multiple analysis of variance (PERMANOVA) model to produce a pseudo-F statistic and a p-value from which to test the hypothesis test for the third research question:

$H3_0$: All coefficients in the multinomial logistic regression equation will take the value of zero.

$H3_a$: The model currently under consideration is accurate in that it differs significantly from the null coefficient values of zero; it gives significantly better than random chance predictive strength relative to the null hypothesis.

The fourth research question involved the concept of self-supervised, unsupervised deep learning models and their ability to determine the presence of human-curated malware classes

without having pre-training on the labeled class identities. This research question served as the cornerstone of a machine learning/artificial intelligence-driven defense-in-depth strategy. My primary research goal was to prove the ability of these ensemble supervised and unsupervised systems to identify novel malware classes without necessarily having to be pre-trained to do so.

This research question addressed, through hypothesis testing, the use of a permuted testing framework based on a PERMANOVA (permuted multiple analysis of variance) a non-parametric test of statistical significance between two groups (benign versus malicious) in a post-hoc test. In this situation, the unsupervised deep learning anomaly scoring model will analyze the network flow data in a blind study configuration (e.g., the human-labeled curated malware class, if any, will be withheld in the data provided to the model). Anomaly scoring took place using the self-supervised/unsupervised deep learning algorithm and network flow observations determined highly anomalous. In turn, the scores were encoded as positive for malware as an outcome prediction.

The original curated network flow dataset used to train the model recoded the binary classification (0/1) variable for the presence or non-presence of a malware event in each network flow. The test of statistical significance determines if the unsupervised model anomaly findings represented by one set of categorical (binary) variables related to the “ground truth” human-curated labeled malware events in the data, and a test of statistical significance were made based on this analysis. A beta dispersion test was conducted between the two groups to generate a visual representation of non-Euclidean centroid distances between the two groups (benign versus malicious) across transformed principal component spaces.

This took place using Anderson's PERMDISP2 procedure for the analysis of multivariate homogeneity of group dispersions (variances). This included the “betadisper” (beta dispersion)

functionality within the Vegan library package, a multivariate analogue of Levene's test for homogeneity of variances. Optimization of non-Euclidean distances between objects and group centers (centroids or medians) through by reducing the original distances to the principal coordinates. The researcher used this procedure to assess beta diversity between distinct groups.

The hypothesis for this fourth and final research question is:

$H4_0$: There is no statistically significant relationship between the predictor variables and the response variable in the anomaly-driven response variable groups.

H4a: There is a statistically significant relationship between the predictor variables and the anomaly-driven response variable groups.

Statement of the Need

Previous research on the security design and unique aspects of Internet of Things (IoT) devices indicated an ongoing challenge in detecting and classifying a growing range of malware threats. Recent studies indicated several static-based methods for IoT malware detection, including analysis of opcode features, string data, ELF headers from binary data files as well as other methods related to the analysis of binary executable machine language code and function calls.

Thus, much of the current state of the art for static-based detection of malware for multi-architecture IoT platforms involves the analysis of machine language instructions and related byte code elements that are already present on the IoT device, presumably after the malware has been installed through a network connection. Several dynamic-based methods for IoT malware detection and mitigation frameworks have also been proposed, including the use of a hybrid combination of machine learning algorithms operating within a Software Defined Networking (SDN)-based secure IoT framework.

While these studies have been significant in moving the state of the art forward, the rapid growth in the adoption and use of IoT platforms connected directly to unrestrained Internet connections necessitates an increased focus on preemptive malware detection and classification for these devices before malicious opcode instructions can be installed and executed by threat actors.

Because of the complex nature of IoT network traffic, coupled with the increasing number of malware attack scenarios and the multi-architecture nature of many IoT devices, multiple modeling techniques are most suitable. Experimentation using a combination of statistical analysis techniques and supervised deep learning algorithms created models for linear and nonlinear feature extraction to perform multinomial classification with probability scoring of the known labeled malware threat classes. From the models produced, a new framework could allow the continued use of IoT devices on unrestrained Internet connections by intercepting IoT network traffic flows that are associated with the presence of a malware threat in real time.

Purpose of Study

The purpose of this study was to conduct research in digital communication systems to effectively utilize machine learning and deep learning methodologies, for the purpose of creating a defense-in-depth strategy for securing IoT devices in enterprise network systems that incorporates self-adapting, nonparametric statistical learning systems. This incorporated aspects of internal and external validity as well as falsifiable hypothesis testing into the research discipline.

The use of novel methods for incorporating statistical hypothesis testing provided scientific evidence for the effectiveness of the supervised as well as the unsupervised learning models, demonstrating the effectiveness of these models against labeled threat malware classes

in unrestrained network connections. The research ultimately explored frameworks and methodologies for establishing scientific validity for the effectiveness of machine learning and deep learning models for an effective use in defense in-depth strategies for IoT hosts in unrestrained network environments.

This study contributes to the academic literature on digital communication systems by addressing critical limitations involving falsifiability and the use of statistical learning systems for IoT malware analysis and cyber defense. Historically, hypothesis testing has rarely been attempted for research that utilized machine learning and deep learning prediction models because the nonparametric nature of these algorithms was often in conflict with linear relationships required between dependent and independent variables for the development of valid inferences for hypothesis testing. Recent developments in "explainable AI" technology frameworks and advances in the use of subsampling techniques and variance estimation, under suitable conditions, can potentially allow for the creation of asymptotically normal predictions that can meet the criteria for some forms of hypothesis testing. The research incorporated these advanced frameworks into this study for addressing the research question hypotheses.

Significance of the Study

The need for adaptable, robust, and scalable real-time malware detection and classification for Raspberry Pi IoT devices in unrestrained Internet-connected networks is apparent, especially given the tremendous growth and near-omnipresence of these devices in industry, hospitals, and home network environments. The most adaptable and scalable technique for detecting a growing number of malware threat classes that are specifically targeted at IoT hardware variations is potentially a robust ensemble deep learning algorithm that is fast enough

to work in near-real time and more accurate than traditional signature-based and anomaly-detection or heuristic-based methods.

The utilization of these methodologies exclusively trained on network flows as opposed to the more detailed packet capture for network traffic analysis, potentially provides a significant enhancement in network defense in-depth strategies. Such an enhancement has the opportunity to optimize significantly network defense in-depth strategies by creating a potentially more self-adapting active malware detection system that is less expensive in terms of storage and processing requirements than statistically based or rules-based malware detection systems that rely on packet capture analysis.

Assumptions

The intent was to develop an ensemble of deep learning classification and prediction models based on various cutting-edge advanced neural networks, for the purpose of classification and probabilistic scoring of known types of IoT malware network intrusion for the Raspberry Pi ARM-based processor hardware. The datasets for the quasi-experiments consisted of hundreds of gigabytes of unencrypted TCP and UDP internet protocol network traffic data freely provided via the Stratosphere Laboratory, using the Aposemat IoT-23 labeled dataset with malicious and benign IoT network traffic. The research used pre-seeded randomization to allow maximized reproducibility within the various predictive classification models, as well as document data sources, base versions of the R statistical programming language, and the versions of all R library packages utilized in this research effort.

Limitations

The primary limitation of this study is that it utilizes the IoT-23 dataset, and thus be an ex-post facto experiment with relatively limited (but still substantial) network flow data. While

this dataset is robust and well-curated, the characterization of findings obtained from the models produced from the relationships in this data may not adequately capture a comprehensive state-of-the-art or fully global perspective on the most modern IoT malware threats.

A broader view of IoT malware adaptability and suitability for enterprise network defense systems that utilize a defense-in-depth strategy as outlined in the research questions for this study could potentially be conducted by obtaining a more comprehensive dataset using controlled networks and honeypot systems than made possible by relying solely on the IoT-23 dataset for ex-post facto research. This dataset capture process might potentially involve the use of network telescopes in partnership with academia, industry, and law enforcement agencies for comprehensively cataloging and curating IoT network traffic flows across the globe, but this methodology would exceed the scope and confines of this research.

Likewise, new methodologies, frameworks, and technologies are constantly being researched and presented in journals and conferences while this study is in progress, it is not possible to incorporate all of these concurrent findings during the course of this research. Thus, the limitations of this study incorporate the scope of the data available to it in the IoT-23 published dataset, and in the algorithms chosen and the methods selected to address the selected research questions.

Key Terms

This research used the following terms for the purposes of this study.

Asymptotic: Refers to the large sample theory in statistics. It is related to the central limit theorem in probability theory. Asymptotic theory states that with a sufficiently large sample size of *n*, that elements drawn randomly from identically distributed and independent (of each other)

variables from the population converged to the overall population mean for those respective variables.

Autoencoding Neural Network: An "autoencoder" is a type of multi-layer perceptron neural network, designed to learn to encode data that does not contain a labeled dependent variable. It is therefore a form of unsupervised learning, often referred to as self-supervised learning because of its use of nonparametric feature reduction to reduce dimensionality as well as provide a reconstruction of the mean square of the error for each observation within a dataset (e.g., a pseudo-dependent variable score). The intended use of the autoencoding neural network (ANN) is primarily to reduce the dimensional features within a dataset to determine statistically insignificant noise from useful signal that exists within the data. It is therefore a potentially efficient mechanism for determining the key response indicators taking place in complex data. Dimensionality reduction via ANNs was one of the first deep-learning applications (Goodfellow et al., 2016). It also provides potentially significant advantages over other legacy unsupervised dimensionality reduction methods such as principal component analysis (PCA), which is based on more traditional, parametric orthogonal transformations (Jolliffe, 2002).

Centroids: The centroids of a dataset refer to the vector mean, typically used in reference to the mean of multiple continuous independent variables in a dataset. Commonly used in clustering methodologies such as k-means, as well as in multiple analysis of variance (MANOVA) and analysis of variance (ANOVA). Judson (2005) refers to the use of the Nearest-Neighbor Centroid Method, using a specific variable chosen for pair matching such as with simple or squared Euclidean distances. Conversely, Mahalanobis distances when utilized are re-scaled Euclidean distances based on the use of standard deviations and accounting for inter-correlation between paired variables. Conversely, Mohanty et al. (2013) discussed the use of

Discrete Fourier Transforms (DFT) when faced with variable feature vectors between objects for classification problems. DFT is able to account for data objects of disparate sizes, utilizing lower-order coefficients to provide the overall shape representation per object, while the higher-order coefficients provide the details for each shape. In this way, a mathematical approximation of the shape features from which to compute centroids.

Categorical Variable: Categorical data is a form of qualitative information in which a given variable is capable of taking on finite values associated with a group of nominal or ordinal-type statistical information. Categorical factor levels refer to each distinct qualitative value within a categorical variable's range of nominal-type observations.

Deep Learning: Deep learning is a subset of machine learning (nonparametric statistical learning algorithms) that utilizes multi-layer perceptron neural networks, in one or more various designs. Deep learning models can perform unsupervised, supervised, or semi-supervised tasks and include various architectures such as multi-layer perceptrons, convolutional neural networks, autoencoding neural networks, recurrent neural networks, and more. While initially inspired by information processing structures in biological organisms, artificial neural networks commonly now deviate from their biologically-based models in order to be more efficient in terms of both understandability and trainability features for specific tasks. The "deep" element of artificial learning neural networks refers to the multiple layers found within all of these neural network architectures that serve as various symbolic representations of nonpolynomial activation functions for enhanced classification capability.

Gradient Boosted Machine: A gradient-boosted machine (GBM) is a supervised learning regression or classification heuristic ensemble algorithm. GBMs are based on the concept of random forests but strive to improve upon the forest algorithm by increasingly refining

approximations of the features contained in the regression trees. This refinement utilizes the principle of gradient boosting within the random forest based on the work of Friedman (2001) which incorporated a functional gradient boost to the original bagging principle of decision trees. This provided the net effect of an iterative gradient descent as a cost function over function space that produces a negative gradient direction that potentially combines a large series of weak learners into a single strong learner in an iterative manner, similar in some manners to a least-squares regression. Gradient boosting has been applied to other areas of machine learning besides random forests, including deep learning neural networks.

Inference: In statistics, inference refers to a methodology of drawing conclusions about a larger population of entities utilizing only a subset of the drawn sample. This method is used to draw some conclusions based on hypothesis testing for this larger population, within a qualified test of statistical significance.

IoT: Internet of Things, refers to technological objects that contain embedded sensors and onboard computer processing ability along with independent network connectivity that allows these objects to communicate over the internet (although not necessarily the public Internet). These objects collectively are often referred to as “smart devices”, such as lighting fixtures, appliances, digital video cameras, etc. as well as hospital healthcare systems. IoT devices allow for a revolution in fields such as healthcare, transportation, factory and home automation, agriculture, military operations, and many other fields due to the ability of these complex machines to be monitored and controlled remotely by advanced software systems in near real-time. Various standards exist to govern IoT usability including addressability, application layers, and wireless and wired communication protocols.

IoT-23: The Aposemat IoT-23 dataset refers to a combined set of unlabeled and labeled datasets consisting of a range of both benign and malicious IoT network traffic flows. It contains both detailed, unlabeled (no dependent variable class) packet capture logs as well as class-labeled human-curated traffic capture flow metadata from twenty distinct types of malware across IoT devices and three types of benign IoT device network flow captures. Stratosphere Laboratory at CTU University in the Czech Republic curated this dataset with human-labeled malware event classifications from 2018–2019 and consists of approximately 21 gigabytes of labeled network flow data. The laboratory executed each of the curated event observations using Raspberry Pi hardware, recording where specific protocols performed specific actions on these devices. The purpose of this dataset was to facilitate machine-learning research into IoT malware phenomena.

Maximal Information Coefficient: The Maximal Information Coefficient or MIC is a measure of the linear or nonlinear relationship between two variables. Reshef et al. (2011) referred to the MIC as belonging to a larger class of statistical measures known as maximal information-based nonparametric exploration (MINE) statistics. The potential advantage of the use of MIC as opposed to a more traditional correlation coefficient such as the Pearson correlation coefficient (commonly referred to as Pearson's *r*) is that the MIC can be sensitive to both linear and nonlinear relationships between two variables.

Network Traffic Flow: Unlike a packet capture log, a network traffic analysis is at a more summary level of detail of packet switching communication activities that have taken place across a digital network. Packet capture logs contain a much greater level of detail, with all portions of a message recorded from origination to final destination as packet segments, combined at the final destination address and allowing a full reconstruction of the original

message from this log file. Conversely, network traffic flows consist of seven distinct attributes that are recorded at the summary statistical level, but do not provide the more expensive and far larger amount of detailed information as exists in the packet capture. Flow analysis is therefore much less expensive (computationally and in terms of storage), but at the cost of information richness.

Nonparametric Statistics: Nonparametric statistics refer to a branch of statistics not primarily based on the concept of a normalized probability distribution. That is, one may consider nonparametric statistical analysis to be distribution-free, or within an unspecified distribution domain. Both descriptive statistics and inferential predictions can potentially be made with nonparametric statistics, utilizing various nonparametric tests including a Non-Parametric ANOVA (NPANOVA) or Permuted Multivariate Analysis of Variance (PERMANOVA), as per Anderson (2001, 2006, 2017). Nonparametric (often referred to as *nonlinear*) model methodologies are also typically determined by their specific datasets, rather than being fixed in advance *a priori*.

Nonlinear Statistics: See Nonparametric Statistics (above).

Neural Network: In statistical learning systems, a neural network is a computational construct that is inspired in principle by biological neural processes that operate with forms of electrochemical stimuli information. Also known as an ANN (artificial neural network), these systems can be applied to a broad range of supervised and unsupervised learning processes, including forecasting (regression and classification) with highly complex data, novelty, or anomaly detection as well as signal filtering and data compression. Many different types of neural network configurations exist, including versions designed for advanced time series regression and object recognition in digital video, audio, or still images.

Machine Learning: Considered a branch of the study of artificial intelligence, machine learning broadly refers to statistical learning algorithms, and more commonly nonparametric statistical learning algorithms. Machine learning algorithms can take the form of supervised, unsupervised, or semi-supervised learning and commonly conduct operations such as regression forecasting, classification forecasting, or data clustering. A subset of machine learning is Deep Learning.

One-hot encoding: With one-hot encoding, numerical representations of each multi-level categorical data factor is represented by a new binary feature ("column") in the dataset to convert this categorical qualitative data to quantitative binary values that can be processed by the machine learning algorithm(s) desired. This of course has the potential to result in a vast explosion of dataset size and complexity when a large number of highly complex qualitative variables are in use with a machine-learning dataset.

Parametric Statistics: Also commonly referred to as "traditional" statistics, parametric models assume that data drawn randomly from a population sample conform to a variety of assumptions of normality associated with aspects of linear regression, including conformity to the family of normal distributions as originally defined by Fisher (1946). Fisher's work cited above forms a majority of the basis for modern, classical linear statistical analysis. Conversely, modern machine learning algorithms typically do not conform to parametric statistical assumptions (see: Nonparametric Statistics).

SHAP Coefficient: Also known as a SHAP Value, SHAP stands for SHapley Additive exPlanations. Based on research from Lundberg and Lee (2017), SHAP Coefficients arguably represent one of the current state-of-the-art in "Explainable AI". Based on research on coalition game theory established in the 1950s by Shapley, SHAP Coefficients are potentially able to

determine marginal contributions from cohorts of independent variables that contribute to nonparametric model predictive strength for a given predictive observation (known as local interpretability) as well as to the collective dataset in the model as a whole (known as global interpretability). Global interpretability for each IV is on a continuous range that is either positive or negative, in order to provide substantial model transparency for the features present. Supervised Learning: Supervised learning (SL) is a machine learning process that utilizes labeled training data (human-curated) where the dependent variable, consisting of either ratio, interval, nominal or ordinal data, is known. This training data can create a predictive model, either a type of classification or regression, to produce an inferred function based on methods utilized within a given nonparametric statistical learning algorithm. Examples of common supervised learning algorithms include random forests and gradient-boosted machines.

Self-Supervised Learning: Self-supervised learning (SSL) is a form of unsupervised learning where the model self-selects the independent variables to be retained in generating a useful prediction. Various mechanisms exist for this to take place, but in all cases, self-supervised learning takes place where unlabeled (no dependent variable) exists or is at least unutilized by the model. Self-supervised learning is often used for nonparametric multivariate anomaly detection as well as some aspects of computer vision, speech recognition, and in some forms of natural language processing such as in Google's BERT algorithm (Xie et al., 2021). Generally, SSL allows for the optimization and use of complex, lower-quality, unlabeled datasets.

Spectral Encoding: Also known as Eigenfactor Encoding, spectral encoding is a methodology for numerically encoding multi-level factor categorical data features into lower dimensional space compared to traditional one-hot encoding methods. Whereas with one-hot

encoding, numerical representations of multi-level categorical factor data are represented by a new binary feature (“column”) in the dataset to convert this categorical qualitative data to quantitative binary values that can be processed by the machine-learning algorithm desired. This of course has the potential to result in a vast explosion of dataset size and complexity when a large number of highly complex qualitative variables are in use with a machine-learning dataset. Spectral Encoding conversely approaches this dataset explosion issue by mapping Laplacian distance measurements between categorical features using Eigen factors in a decomposition matrix for each categorical independent variable. This results in a necessary quantitative conversion of categorical features in a much more efficient relationship matrix for each categorical feature without loss of statistical information or the far more cumbersome and computationally expensive one-hot feature encoding process.

Unsupervised Learning: As opposed to a supervised learning methodology, unsupervised learning utilizes unlabeled datasets, which are structured datasets that contain no labeled dependent variable. As such, unsupervised algorithms rely on self-organization based on a combination of statistical features and various types of probability densities in order to determine relationships in the dataset. Examples of probabilistic algorithms include k-means clustering and DBSCAN clustering (Density-based spatial clustering of applications with noise), and unsupervised anomaly detection methods such as isolation forests, or principal component analysis, which is an unsupervised dimensionality reduction technique based on orthogonal linear transformations.

CHAPTER 2

REVIEW OF LITERATURE

Bodies of Knowledge in Digital Communication Systems

IoT devices are largely computationally resource constrained and have access to relatively little onboard memory, while also contending with small power supplies due to constrained energy storage, diminutive power distribution, and restrictive power scavenging capabilities (Raj & Steingart, 2018). Likewise, some of the challenges related to malware detection and classification for Internet of Things (IoT) devices using a network intrusion detection system (NIDS) is that IoT network traffic largely consists of homogenous protocols, hardware, and software components (Anthi et al., 2019).

Bobrovnikova et al. (2019) stated that the vast (and rapidly growing) number of IoT devices, inherently poor network security processes, and their typical reliance on unrestrained permanent Internet connections have made IoT devices a convenient tool for threat actors to infect these devices and then organize them for powerful cyberattacks. Murphy (2017) expanded on this by discussing how many IoT device manufacturers are focused on enhancing profitability by making these devices as inexpensive and quickly manufactured as possible and sacrificing many security elements in their design as a result.

Based on Garcia et al. (2020), the IoT-23 dataset is a dataset based on curated, human-labeled network flow data that consists of internet protocol network traffic flow observations

from Internet of Things (IoT) devices. These IoT devices are all based on the Raspberry Pi hardware architecture standards as applied to a range of specific consumer electronic devices that were connected to network host devices.

The dataset itself contains a mixture of 20 curated malware captures that took place on these IoT devices as well as labeled benign network flow traffic, plus three specific sets of 3 captures for benign IoT devices traffic. First released in the first quarter of 2020, this curated dataset included network flow traffic that dates from the timeframe of the years 2018 and 2019. The Stratosphere Laboratory and the AIC group within CTU University of the Czech Republic then produced the curated IoT network flow data for future utilization in the quasi-experiments.

The stated objective for this human-labeled dataset is to provide a substantial collection of actual, organized, and classified IoT executable malware infestations with simultaneous IoT benign-classified network flow traffic for scientific research to potential detection and mitigation strategies, primarily utilizing artificial intelligence and machine learning algorithm-driven adaptable software solutions. Avast Software in Prague, Czech Republic provided the funding to capture and organize this dataset collection.

According to Avast Software via their partnership with the Stratosphere Laboratory, “The IoT-23 dataset consists of twenty-three captures (called scenarios) of different IoT network traffic. These scenarios comprised twenty network captures (pcap files) from infected IoT devices (which had the name of the malware sample executed on each scenario). Three additional network captures provided real IoT devices network traffic that have the name of the devices where the traffic were produced.” The Laboratory executed a specific malware sample for each malicious scenario on a Raspberry Pi that used several protocols and performed different actions.

The Laboratory further stated that “Table 1 shows the characteristics of the IoT botnet scenarios and Table 2 shows the protocols that were found in each network traffic capture. The network traffic captured for the benign scenarios obtained by capturing the network traffic of three different IoT devices: a Philips HUE smart LED lamp, an Amazon Echo home intelligent personal assistant and a Somfy smart door lock. It is important to mention that these three IoT devices are real hardware and not simulated. This allows us to capture and analyze real network behavior. Both malicious and benign scenarios run in a controlled network environment with unrestrained internet connection like any other real IoT device. Table 3 shows the network data of the IoT benign scenarios and Table 4 shows the protocols found in each network capture.”

The primary type of IoT hardware platform utilized in this series of deep learning AI quasi-experiments for data capture and malware validation was the Raspberry Pi IoT device type, utilizing extensive third-party datasets of Internet network traffic data (Parmisano et al., 2020) for both benign and permutations of malware infestations on these devices. The quasi-experiments utilized network traffic flow captures exclusively, to utilize inexpensively obtained, fulfilling a need to manage data storage volumes and cost requirements for available resources. In this research, the author addresses how network flow data may potentially contain sufficient attributes and variabilities for identifying potential malware events, while not relying on more costly packet capture storage and analysis systems.

The Raspberry Pi Foundation GitHub (2020) documentation defined the Raspberry Pi as a series of small, inexpensive single-board computers, originally designed in the United Kingdom. Specifically, the Avast AIC laboratory utilized the raspberry pi to execute the malicious malware samples to validate the presence of threat actor malware.

Originally designed as inexpensive basic computer science aids in schools and developing countries soon outsold its original target market. By 2016 Raspberry Pi had sold over 10 million units (Raspberry Pi Foundation, 2016). As of 2020, the Raspberry Pi platform has become a ubiquitous IoT device type, being a significant component in the overall IoT global device landscape which Hung (2017) estimated would be approaching 20 billion networked devices by 2020. With a wide range of diverse networked IoT applications including greenhouses and hydroponics (Lukito & Lukito, 2019), energy monitoring (Mudaliar & Sivakumar, 2020), health monitoring (Ganesh, 2019), smart traffic monitoring (Mehta & Patel, 2019) and smart infant monitoring systems (Ibrahim et al., 2019), the Raspberry Pi represents a rapidly growing component within a vast and flourishing population of autonomous Internet-connected devices.

The consequences of malware anomalies attacking or infecting IoT devices including Raspberry Pi-based IoT are significant. Gupta et al. (2015) discussed a proposal for healthcare-based IoT devices for use in monitoring patient Electrocardiogram (ECG) as well as for other vital statistics. Authorized personnel could review data flows identified as potential risk elements in real-time over the Internet for storage and further analysis. Likewise, Jaiswal et al. (2017) discussed the use of Raspberry Pi-based IoT sensor devices to monitor various patient symptoms in digital format and then transmit this data to an Internet gateway through Bluetooth wireless connections to a cloud storage service.

Such standardized, low cost and ubiquitous devices have substantial potential revolutionary benefits for healthcare and other critical fields by allowing real-time data gathering and transmission of a wide variety of critical data streams. Human subject matter specialists as well as additional AI and autonomous diagnosis platforms (Keane and Topol, 2018) would then

provide more detailed reviews. Thus, Raspberry Pi IoT devices suspected of network-based malware attacks can potentially degrade or even suspend their data gathering and data transmission processes, or even potentially alter the content and destination of their datasets, with potentially catastrophic results to a large number of human patients.

It is thus clear that a need exists to explore robust solutions that are specific to a subset of the IoT market based on heterogeneous hardware and operating system types. This can provide substantial utility in that it may provide a mechanism to both accurately and rapidly detect as well as to classify with high probability distinct malware intrusion events in order to help mitigate and prevent potentially catastrophic system damage. Further, the ability to detect unknown (non-human labeled) malware classes that may be novel to the heterogeneous hardware was explored and demonstrated, as part of a robust defense in-depth strategy.

Some of the potential advantages of using machine learning (and its more capable cousin, deep learning) over "classical" statistical classification techniques such as logistic regression are discussed by Bzdok et al. (2018). The authors state that one key difference is that classical statistics draw inferences from populations, whereas machine-learning algorithms attempt to find generalizable predictive patterns. The authors further state that statistical models are often based on this inference, which is achieved through the creation and fitting of a project-specific probability model. The (statistical) model allows the practitioner to compute a quantitative measure of confidence where a discovered relationship describes a 'true' effect that is unlikely to result from noise. Thus, when enough data is available, the practitioner can explicitly verify assumptions (e.g., equal variance) and refine the specified model as needed. By contrast, ML (machine learning) concentrates on prediction (or classification of multinomial classes) by using general-purpose learning algorithms to find patterns in often rich and unwieldy data.

ML methods are particularly helpful when one is dealing with ‘wide data,’ where the number of input variables exceeds the number of subjects. It is therefore proposed in this research study to utilize the most recent advances in machine learning, i.e. neural network deep learning algorithms, for use in classifying complex malware behaviors in exceedingly complex and wide network traffic capture datasets to solve a problem that traditional signature-based and similar statistical anomaly detection and rules-based models may not adequately address.

This research used Stratosphere Laboratory datasets in developing the proposed analytical models from the quasi-experiments. Stratosphere Laboratory developed their Aposemat IoT-23 dataset of Raspberry Pi IoT malware attacks in order to facilitate this type of research (Parmisano et al., 2020). Their Raspberry Pi labelled datasets include malware attacks from a wide range of 20 different attack types on unrestricted Internet-connected Raspberry Pi IoT devices, as well as 3 sets of network traffic captures from benign IoT device traffic. The lab produced a significant and robust dataset of network traffic captures beginning in 2018 and continuing through 2019 in the Czech Republic. Their datasets are divided into 23 different scenarios consisting of different types of IoT network traffic, including 20 from malware-infected network devices and three captures of Internet connected devices from outside of their lab that have the name of the devices where the traffic was captured from.

Some of the most recent prior art for using machine learning and artificial intelligence techniques for IoT anomaly detection include Bobrovnikova et al. (2019), which used two data sets in their experiments, the BoTIoT dataset (Koroniotis et al., 2018) and the UNSW-NB15 dataset (2020). For their specific use case and dataset scenarios, these authors proposed the use of semi-supervised fuzzy c-means clustering, SVM, and Artificial Immune System classification algorithms, by focusing on elements of DNS traffic of the IoT network and relying on “white” vs

"black" domain name listings. However, techniques including DNS spoofing techniques as discussed by Sivaraman et al. (2018) may compromise aspects of some of these narrowly focused IDS approaches and limit their inherent flexibilities.

The explosive growth of IoT machines coupled with the ubiquitous presence of the Raspberry Pi platform that is easily adaptable for commercialization as an IoT product has led to a significant homogenization for industrially manufactured IoT devices. The increasing use of IoT machinery within unrestrained network connections that provide direct Internet connectivity, coupled with homogeneity and increasing mission-critical device applications such as life support monitoring requires the presence of scalable, adaptable malware detection and classification in order to provide threat mitigation.

Multiple unique characteristics exist for IoT devices and in particular the Raspberry Pi and the real-time, potentially vulnerable and bidirectional streaming characteristics of IoT data streams distinct from traditional Internet traffic such as web browsing (HTTP, HTTPS protocols) and email (SMTP protocol, etc.). Coupled with the fact that over 98% of IoT network traffic is presently unencrypted (PaloAlto Networks, 2020), leads to the issue that traditional signature-based and algorithmic-based malware detection and classification algorithms for network intrusion/anomaly detection are largely inapplicable (McCarthy, 2020).

McLean and McLean (2001) defined HRD comprehensively. Their defining quote was when they stated that "Human resource development is any process or activity that, either initially or over the long term, has the potential to develop adults' work-based knowledge, expertise, productivity and satisfaction, whether for personal or group/team gain, or for the benefit of an organization, community, nation or, ultimately, the whole of humanity."

Gilley et al. (2002) discussed the definition of HRD in 2002. In their seminal study, they stated that HRD is about "the effective development of people within an organization, and specifically differentiates human resources from physical resources such as machines, facilities, materials, equipment and component parts of products, or what would often be referred to as fixed corporate assets as well as financial resources within an organization."

Thus, human resource development seeks to improve upon the overall knowledge, competencies, skills, and attitudes of the human members within an organization. The application of the proposed research models ultimately results in several potential benefits. Firstly, it provides a proposed enhancement for overall network "defense in depth" for IoT data against malware classes. Secondly, it demonstrates an ability to detect novel malware classes in network flows without prior training. It also provides a better understanding of overall IoT malware risk within the enterprise that is applicable through principles associated with human resource development (HRD). Finally, the application of this research provides a novel foundation of scientific falsifiability that proves the effectiveness of these methods, in addition to utilizing various traditional measurements of AI/ML model effectiveness.

Aspects of this research directly address the ability of the research to be "explainable" to address the HRD component of the organization that governs the use of the Raspberry Pi networked devices to facilitate the understanding of the impact and risk components produced by those devices for the larger enterprise.

According to Woolman and Lunsford (2022), the increasing commonality of threat actors, coupled with ever-increasing network complexities and the growing sophistication of threat tools greatly increases the vulnerabilities of critical network infrastructure and host systems more than ever before.

The authors state that the ability of potentially targeted enterprise organizations and government entities to defend their network perimeters utilizing traditional threat detection systems provides only a limited set of tools, traditionally based on simple, and typically parametric, inflexible statistical tests of network activities and known threat signatures. These threat signatures generally rely on pre-defined malware detection rules based on known, previously encountered network intrusion classes. As a result, vast resources of confidential information, sources of competitive advantage, national security information as well as critical resource applications can potentially be highly susceptible to an increasingly novel suite of evolving network intrusion types that are likewise increasing in frequency.

This potentially puts a vast range of commercial and public sector resources and information in mounting jeopardy, particularly in the domain of IoT network traffic is often more vulnerable than non-IoT device data. Non-IoT data typically originates from such devices as mobile phones, tablets, PCs, and laptops. These devices often have more capable onboard processing power than IoT devices, allowing for some fundamental malware detection including independent software firewalls and robust encryption capabilities.

The detection of cyber threats from known legacy malware risks utilizing a multi-layered defense approach based on pioneering research from Kephart, Sorkin, Chess and White (1997). Their work in turn utilized the transformation of signature detection processes as proposed by Cohen (1987). While signature-based network malware and intrusion detection is still among the most heavily used techniques, heuristic approaches that are able to discern multiple, related threats from a single definition source have been increasingly common as defined by Kaspersky Lab ZAO (2013).

However, novel threats as well as more advanced cyber malware and intrusion events that are specifically engineered to avoid detection by the more commonly used available tools and techniques are becoming increasingly common. By being able to bypass the network security perimeter, intrusions and malware can easily propagate throughout the network and operate undetected for substantial lengths of time. In many cases, these network intrusions can access restricted information while remaining undetected, masquerading their traffic signatures as legitimate, benign activities.

As the capability to resist successful classification is increasing with the latest generation of network intrusion technologies, continuous improvement in the multi-layered network defense approach first proposed in 1987 becomes increasingly necessary. One example of this emerging malware threat class is a sophisticated piece of modular malware known as Flame, first discovered in 2012 on networked devices running the Microsoft Windows operating system (ICIRT, 2012). Flame, also known as Skywiper, was likely created by a state actor as a cyber-weapon deployed for espionage purposes for a target in the Middle East (Kaspersky Lab ZAO, 2013).

As discussed in Woolman and Lunsford (2022), the discovery of Flame occurred circa 2012. Now generally regarded as an unusually robust backdoor attack toolkit, Flame also contains substantial worm-like features and Trojan malware capabilities. Among its unique capabilities is the ability of Flame to replicate within a targeted network as well as on removable media upon receipt of commands to do so by a remote threat actor's command and control server. While the exact method of entry into a network has not yet been determined, Flame's ability to take on different roles through a wide range of add-on functional libraries allow it to be extremely adaptable and difficult to analyze by traditional mitigation and detection methods. One

particularly unique attribute of Flame was the utilization of the novel technique of concealment through an unusually large and variable codebase compared to most other network malware threats.

The authors discussed how Flame is capable of harvesting sensitive data in a variety of ways, including robust SQL database query insertions, compressed digital audio microphone recording, Bluetooth wireless connectivity attacks from inside the network, as well as file and network traffic ingestion and analysis. Flame can also take recurring screenshot images from infected devices. Flame is capable of reporting to an external command and control server from within the targeted network via a covert SSL data channel, as well as turning other host devices within the network into beacons that are discoverable via Bluetooth connections, according to Kaspersky Lab Zao (2013).

Advances in these emerging categories of malware and network intrusion capabilities thus require adaptable learning technologies for detection that are not based on pre-defined statistical patterns, heuristics, or rule-based detection methods. One increasingly utilized form of malware and network intrusion detection is anomaly-based detection methods, often utilizing data mining technologies including machine learning and deep learning. Deep learning anomaly detection utilizing network traffic analysis such as packet capture and network flow data is one such emerging advance in this field.

An advantage of utilizing packet capture datasets for anomaly-based network intrusion detection systems (NIDS) is that full packet capture allows for a mirror image of the entirety of the network traffic for a given time period, allowing robust deep packet inspection (DPI). The DPI data allow for a full forensic analysis of all available features including protocols, payloads,

and source and origins for each packet, as well as a variety of measurements related to packet transmission speeds and delays.

One disadvantage of packet capture dataset forensics for NIDS is that DPI imposes a significant burden on routers, switches, and network infrastructure in general during this mirroring cycle to capture and store the vast amount of network traffic. Further, the data storage, processing, and analysis of these are often quite deep (often multi-terabytes per day within enterprise networks), wide (typically on the order of dozens of independent variables per observation), and complex datasets often require the use of more cumbersome "Big Data" analytical cluster computer environments. These specialized analytical frameworks thus necessitate an increase in the scope and complexity of these projects. Robust encryption methods of packet payload data further increase the signature detection complexity of DPI analysis (Woolman & Lee, 2020).

Network flow datasets represent a more "high level" metadata scope of network traffic within the enterprise, providing summarized level data between the source (IP and port) and destination (IP and port) per protocol. Rather than recording the actual packet payload of each component of network traffic, the network flow dataset typically records only the information about the number of packets per flow observation, as well as the Shannon entropy for each packet payload, and the duration of each flow.

Because network flow provides only the "headlines" of the more complex network traffic, it is able to provide a succinct, high-level view of activities taking place across the network including timestamps. These headlines incorporate the IP (internet protocol) addresses and ports for both sender and receiver, the overall length of the conversation, the protocol used, the amount of data sent, and an estimation of the amount of information potentially held within

that data. This summarized view of network traffic substantially reduces the burden of storage, processing, and analysis of enterprise network traffic, at the cost of reduced granularity of the dataset. A full recorded network history using DPI would be required to provide a fully detailed forensic review of network traffic for a given time period.

Due to the more concise nature of network flow data, it provides a more cost and computationally efficient approach for NIDS that is potentially closer to real-time for network intrusion detection. However, the decreased granularity of this dataset requires a sophisticated analysis technology that is adaptable to ever-changing network traffic patterns, minimizes false positives and false negatives (Type I and Type II statistical errors), and does not rely on pre-defined signature rules, heuristics, or statistical pattern definitions. Thus, machine learning and deep learning technologies come heavily into focus with this research.

Table 1 (below) indicates attributes from the curated network flow data, providing a summary analysis of all 20 known malware classes known to be present within the complete universe of Iot-23 dataset elements. As provided by Stratosphere Laboratory, a summary analysis provided aggregate count information for the number of packets and network flow observations (“ZeekFlows”; Parmisano, 2020). Likewise, the lab provides an aggregate amount of packet capture data related to each malware class in the total Iot-23 dataset, the total duration in hours of observation time for each distinct observation period, and the attack class labels encountered in each observation window (not included benign labeled activities).

Stratosphere Laboratories (“Stratosphere”) obtained these attributes by running the Zeek network analysis framework on the original pcap file, with the size of the original pcap file and the possible name of the malware sample used to infect the device being determined through their meta-analysis.

Because the malware captures executed over long periods of time and due to the large size of the traffic generated by each infection, Stratosphere rotated the pcaps files generated by once every 24 hours. However, in some cases, Stratosphere reported that the pcap (packet capture) log file was growing too fast and therefore it decided to stop the capture before the twenty-four hours window elapsed. For that reason, some of the pcap capture logs differ in the hours used for a given time window.

Table 1

Curated Malware Attack Classifications Labeled in the 20 Distinct Dataset Capture Sessions

#	Name of Dataset	Duration (hrs)	#Packets	#ZeekFlows	Pcap Size	Name
1	CTU-IoT-Malware-Capture-34-1	24	233,000	23,146	121 MB	Mirai
2	CTU-IoT-Malware-Capture-43-1	1	82,000,000	67,321,810	6 GB	Mirai
3	CTU-IoT-Malware-Capture-44-1	2	1,309,000	238	1.7 GB	Mirai
4	CTU-IoT-Malware-Capture-49-1	8	18,000,000	5,410,562	1.3 GB	Mirai
5	CTU-IoT-Malware-Capture-52-1	24	64,000,000	19,781,379	4.6 GB	Mirai
6	CTU-IoT-Malware-Capture-20-1	24	50,000	3,210	3.9 MB	Torii
7	CTU-IoT-Malware-Capture-21-1	24	50,000	3,287	3.9 MB	Torii
8	CTU-IoT-Malware-Capture-42-1	8	24,000	4,427	2.8 MB	Trojan
9	CTU-IoT-Malware-Capture-60-1	24	271,000,000	3,581,029	21 GB	Gagfyt
10	CTU-IoT-Malware-Capture-17-1	24	109,000,000	54,659,864	7.8 GB	Kenjiro
11	CTU-IoT-Malware-Capture-36-1	24	13,000,000	13,645,107	992 MB	Okiru
12	CTU-IoT-Malware-Capture-33-1	24	54,000,000	54,454,592	3.9 GB	Kenjiro
13	CTU-IoT-Malware-Capture-8-1	24	23,000	10,404	2.1 MB	Hakai
14	CTU-IoT-Malware-Capture-35-1	24	46,000,000	10,447,796	3.6 GB	Mirai
15	CTU-IoT-Malware-Capture-48-1	24	13,000,000	3,394,347	1.2 GB	Marai
16	CTU-IoT-Malware-Capture-39-1	7	73,000,000	73,568,982	5.3 GB	IRCBot
17	CTU-IoT-Malware-Capture-7-1	24	11,000,000	11,454,723	897 MB	Linux, Marai
18	CTU-IoT-Malware-Capture-9-1	24	6,437,000	6,378,294	472 MB	Linux, Hajime
19	CTU-IoT-Malware-Capture-3-1	36	496,000	156,104	56 MB	Muhstik
20	CTU-IoT-Malware-Capture-1-1	112	1,686,000	1,008,479	140 MB	Hide and Seek

Note. A randomized selection of these attack class observations used as a portion of the training, testing, and cross-validation data.

CHAPTER 3

METHODOLOGY

The author conducted the research utilizing the R (R core team, 2019) statistical programming language and the RStudio development environment (RStudio Team, 2020), along with a wide range of open-source, deep learning package library algorithms. Data science experiments for the classification of labeled datasets for distinct known malware entities and known benign IP network traffic from Raspberry Pi IoT devices were conducted using a variety of deep learning ensemble algorithm combinations, cross-validated to ascertain the level of accuracy. The research explored the effectiveness of the use of balanced classifier data that is statistically derived using combinations of oversampling, undersampling, and synthetic data techniques as to their effectiveness on the research questions.

Nonparametric Diversity Analysis

Reproducibility for this research relies primarily on the use of the Vegan package for R as discussed in Oksanen et al. (2013) that primarily features peer-reviewed tools for nonparametric diversity analysis as well as ordination methods and methods for dissimilarity analysis. Specifically for this research, the author employed the adonis2 function within the Vegan library package to conduct permuted multivariate analysis of variance using distance matrices. This incorporates a permutation test, and in this case, I used 200 permutations (as a

generally accepted allowable convention as well as a limitation due to the computational expense related to permuted tests of such a large dataset).

Subsampling for Computational Economy with PERMANOVA

The research used a Chi-square Goodness of Fit Test to determine how a randomly generated n of IoT-23 flow observations from the full available population can provide sufficient statistical power for small effect size measurement using the dependent variable categorical factors present. This random sample population allowed for the economization of data to process by the adonis2 function within the Vegan package due to available computing resources, while still enabling sufficient statistical power for detecting small effect sizes with the number of dependent variable factor classes encountered. The process to produce a visualization of the dependent variable group dissimilarities began by first calculating the Bray-Curtis distances between groups using the “vegdist” (vegan distances) function in the Vegan package. A Bray-Curtis distances method was used to determine the multivariate dispersion between groups using the “betadisper” (beta dispersion) function supplied by Vegan, to calculate multivariate dispersions across principal component axes.

The results from “betadisper” (beta dispersion) function provided the multivariate centroids for each group within the dependent variable in the random subsample dataset, producing the Homogeneity of Multivariate Dispersions table shown in the results. This table shows the positive and negative Eigenvalues across each principal component axis used to generate the Beta Dispersion plot between groups using the non-Euclidian Bray-Curtis (“Bray”) distances.

Inter-observer Reliability for Multi-level Categorical Factor Data

According to Szepannek (2022), the R statistical programming environment provides a popular choice for statistical analysis methodologies, in particular meta-analysis, which the authors define as the synthesis of effect sizes from multiple primary studies. Weber (2004) stated that open-source software, which includes the R and RStudio statistical programming environments, represents a significant breakthrough not just in technology but in fundamentally re-framing what constitutes property. Weber further stated that open-source software recasts the notion of some of the basic problems of governance, all to further facilitate the development and use promotion of high-quality, peer-reviewed scientific software at a more rapid and innovative pace than has historically been possible.

This research utilized a robust range of open-source software based on the R statistical programming environment with the objective of producing high-quality research in a short period of time and at a minimal cost than would have historically been possible otherwise.

The Role of Nonparametric Statistical Learning Algorithms

McAlexander and Mentch (2020) addressed the use of nonparametric, nonlinear machine learning, and artificial intelligence models for use in hypothesis testing. Their methodology incorporated research that has indicated that under specific conditions with regard to regularity, it is possible to utilize a means-based process across subsampled predictions from the same model to produce asymptotically normal (linear) predictions from classical statistics based on the general linear model (GLM). Thus, this exploitable process can generate both hypothesis tests as well as confidence intervals that are equivalent to those generated directly from within a traditional, parametric framework. The potentially added benefit of the methods proposed by these authors is that the potentially more subtle nonlinear relationships from the machine

learning and deep learning models can be uncovered in the hypothesis-testing framework, thereby providing a new perspective on the ongoing research topics.

Nonparametric Tests of Statistical Significance for Falsifiability

Anderson (2017) discussed the use of PERMANOVA as a nonparametric ANOVA equivalent, also known as “Permuted Multivariate Analysis of Variance”, that also incorporated the support for pair-wise *a posteriori* comparisons between levels for single factors for categorical variables. This methodology incorporated the use of individual levels of other independent categorical factor variables for the use case of significant interactions and the use of correct permutable units in each case. Anderson’s methods incorporated one-way designs of experiments, multi-factor designs, and Monte Carlo simulations to support permuted p-values to reproduce asymptotic, rigorous statistical inferences from nonparametric model interactions.

This research made substantial use of this subsampling asymptotic transformation method to establish the use of hypothesis testing for a number of the research questions in this research, to establish falsifiable findings for malware events utilizing IoT-23 network flow data.

Tuck and Boyd (2022) discussed the use of Eigen-stratified models for categorical feature encoding for K values within a model, to dramatically reduce the size of the machine learning or deep learning models when held in memory. Traditional feature encoding for multilevel factor encoding for categorical independent variables includes one-hot encoding as well as Laplacian-regularized and stratified models, but either of these methodologies increases the model size to a minimum of K multiplied by the number of independent variables (IV) of the model, which can be substantial and potentially impractical for many computer hardware platforms.

According to Stuber, Chizinski, Lusk and Fontaine (2019), the PERMANOVA methodology as encapsulated by the adonis function of the vegan package (Oksanen et al., 2013)

employs a calculation of the transformed centroid of the sequentially combined independent variables and then determines the squared deviation for each distinct group factor ("treatment group") within the dependent variable to that centroid. The methodology then conducts tests of statistical significance utilizing a pseudo F-test from the sequential sum of squares, based on permutations of the available raw data sample set provided.

Multi-Layer Feed Forward Artificial Neural Networks

The research described in this paper largely utilized open-source software packages including H2O, including the Deep Learning Architecture that consists of "multi-layer, feedforward neural networks for predictive modeling." The author discusses the implications of the activation and loss functions used as well as the regularization methods applied.

The research used a deep learning neural network to create a multinomial predictive classification model that incorporates stochastic gradient descent and incorporates back-propagation to enhance the non-parametric statistical learning capacity for the model. Back-propagation allows for the creation of a gradient loss function for the neural network based on variable weights determined by the neural network algorithm. The back-propagation utilized a gradient descent variant known as stochastic gradient descent for training the proposed three-layer neural network, calculating each layer of the network in sequence and iterating backward from the previous layer to improve efficiency by not re-calculating prior terms from the chain.

Goodfellow et al. (2016) discussed the advantages of back-propagating neural networks, stating that back-propagation is a mechanism referring to the process of computing the gradient within a loss function for machine learning. Whereas the actual learning (convergence) algorithm itself, which in this case is stochastic gradient descent, is what is specifically used to perform learning using the gradient produced by the network. According to the authors, back-propagation

is often misunderstood. Often considered as a specific feature found only in multi-layer neural network learning models, yet in principle, back-propagation can compute across a wide variety of function types.

The initial predictive classification model employed to address Research Question 3 (RQ3) was the H2O Deep Learning algorithm, as discussed in Candel et al. (2016), and employed within the R statistical computing environment. The author used a grid-search parameter optimization process to determine the optimal parametric weights, layers, loss functions, and other parameters employed by this neural network predictive model.

As a convenience for time and available computational resources, the author used a statistical multinomial oversampling process to balance the available representations for all of the available malware classes, including the labeled benign classes in the IoT-23 dataset, limited the available dataset used by the deep learning predictive model to n rows of statistically balanced data across all classes. The researcher based the n of randomly selected observations from the total population on the result of a Chi-square Goodness of Fit Test, for measuring small effect sizes with 23 distinct classes in the data and determining recommended minimum sample sizes for a qualitative (multi-level factor) dependent variable.

This allowed for the production of a minimal sample size estimate using the Chi-square model parameters of a significance level (α) for our defined Type I statistical error, for measuring a “small” effect size (w), with a to-be-defined statistical power of at least 0.8, for 23 distinct factor levels within the dependent variable. The sample size estimate produced by this Goodness of Fit test resulted in an n, randomly sampled from the total population of available network flow observations, across all malware classes. That sub-sampled n dataset gained the

statistical power used for training our deep learning (multi-layer) feed-forward predictive model for RQ3.

To enhance the conservative nature of this approach, the researcher selected this n from the total population using class balancing with the “dplyr” (deployer function) library package in the R statistical computing environment, as discussed in Broatch et al. (2019).

The author then used these n oversampled, randomly selected observations with the h2o library package deep learning, multi-layer perceptron neural network algorithm. This provided a framework with which applied hybrid over-sampling and under-sampling of the minority and majority classes (as defined by the response in the training data) to re-balance the response classes as required to minimize overfitting bias in the predictive model.

This resulted in an internal h2o neural network data frame sized larger than the initial data frame subset from the IoT-23 population. This was due to the oversampled data frame within the neural network contained original and synthetic balanced data that retained the features and data patterns present within the original data, to prevent bias in the model training data that likely favored over-represented class observation in the training data.

As discussed by Banerjee et al. (2018), failing to adequately balance training data is likely to result in the model failing to sufficiently “learn” about the patterns present in the predictor variables as they relate non-parametrically to the response variable classes. Rather, the model in such instances may learn to “guess” about class membership in an observation by learning what the majority class labels were in the data. By using statistical class balancing in our post-hoc training data, the author proposed to address the “lazy classifier” issue present in many gradient descent and tree-based learning algorithms.

This research used adaptive learning in the deep learning neural network to avoid inefficient, slow model convergence within the stochastic gradient descent process within the three hidden neural network layers. The management of this adaptive learning rate took place within the h2o library package for the R statistical computing environment. It used the ADADELTA algorithm as discussed in Zeiler (2012). The ADADELTA algorithm functions within the constraints of the h2o deep learning neural network to pool the benefits of the learning rate annealing with momentum training during the stochastic gradient descent.

SHAP Coefficients for Deep Learning Explainability

The author discussed the potential AI/ML explainability features from the proposed predictive model(s), using various criteria such as predictive model key response indicators (KRIs) as well as SHapley Additive exPlanations values. First proposed as a credit allocation technique between participants in cooperative game theory (Shapley, 1953), the methods originally proposed have since been adapted to apply to predictions from “black box” AI/ML predictive models. The methodology employed by FastSHAP employs a single-pass explainer model that utilizes stochastic gradient descent as the objective function. The enablement of this process is by a value weighted least-squares framework that produces a relatively fast and efficient computation of a gradient outcome optimization.

The purpose of applying Shapley additive features is in applying these metrics to principles associated with human resource development (HRD) within the constraints of technology management. By applying aspects of the predictive classification model KRIs to the use case of malware detection and classification within unrestrained network connection IoT network traffic flows, the research hoped to produce knowledge about the predictability of malware classes in IoT data applied to HRD for the benefit of the technological enterprise. The

research explored such an application by using attributes of explainable AI feature analysis methodologies, including SHAP values.

In-Memory Compression for Qualitative Ordinal Data

Candel, Parmar, LaDell and Arora (2016) discussed the use of an open-source analytical software package ("H2O") that utilizes in-memory dataset compression, allowing it to handle many billions of rows of data in memory, thus potentially making it a useful package of complex IoT machine learning analytics. Those authors stated that this package implements best-in-class (machine learning) algorithms at scale, including gradient boosting and deep learning models within the R statistical programming environment, thus "nurturing a grassroots movement of physicists, mathematicians, and computer scientists to herald the new wave of discovery with data science by collaborating closely with academic researchers and industrial data scientists."

By incorporating Eigenfactor encoding, also known as spectral encoding, the author minimized the storage and processing of multi-level factor categorical data from the IoT-23 dataset network traffic flows in the unsupervised and supervised prediction models. The potential benefits derived include retaining only K columns per categorical independent variable feature that has a demonstrable linear or nonlinear relationship for a given IV. Therefore, this method allowed for the retention of the projections (transformations) of each of the categorical IVs onto a K-dimension eigen space only, through a transformation to a matrix of distinct continuous values for each feature relationship in the eigen space for each categorical IV. This can dramatically reduce the size and improve the computational performance of the model, which can be especially critical in near real-time cyber-defense use cases.

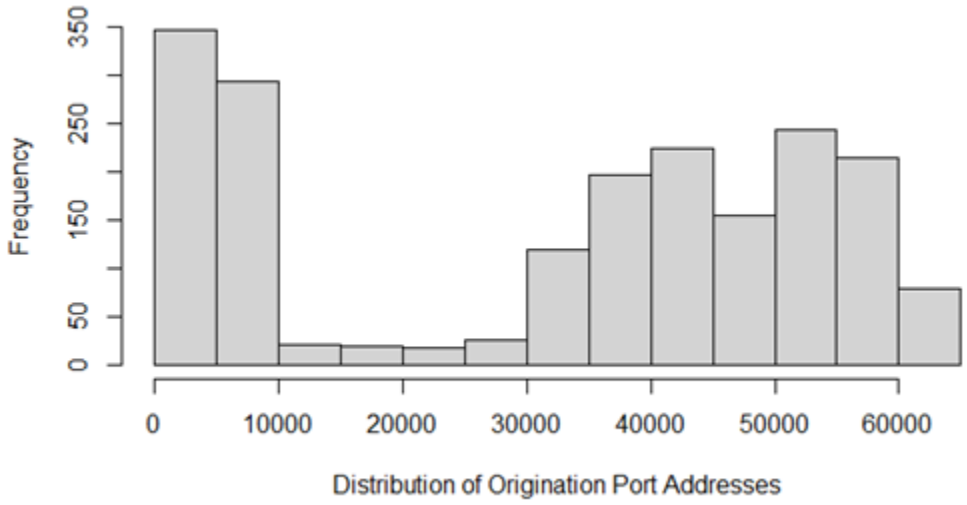
The research produced Shapley value estimations for AI/Ml feature explainability using the FastSHAP library package as developed by Jethani et al. (2021). FastSHAP proposes to

address the computational time expense in explaining “black box” models that involve large, high-dimensional models. FastSHAP utilizes a single-forward pass approach that incorporates a learned explainer model. To inexpensively (from a computing resource perspective) process training without pre-labeled curated Shapley variable scores, FastSHAP incorporates a stochastic gradient descent that makes use of an efficient least-squares process that utilizes variable weights. The resulting objective function provides accurate explanatory cohort value explanatory sets in tabular datasets with an order-of-magnitude processing time improvement over many previous Shapley implementation methods.

Analyzing the distribution of several independent variables of potential interest as key response indicators within this single-class dataset produced distributions as follows, using the origination (sender’s) internet protocol (IP) port address, as shown in Figure 1 (below):

Figure 1

Network Flow Origination Port Address - Honeypot Captured Network Flows for Curated Benign Class

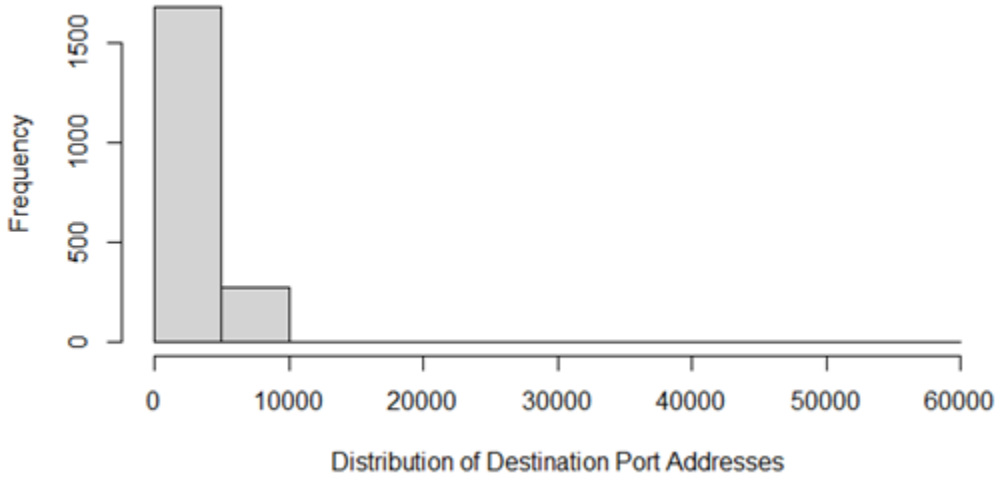


Note. The distribution of network flow origination port addresses

Figure 2 (below) shows the distribution range of internet protocol (IP) port addresses used for the destination in this set of network flows.

Figure 2

Network Flow Destination Port Addresses - Honeypot Captured Network Flows for Curated Benign Class

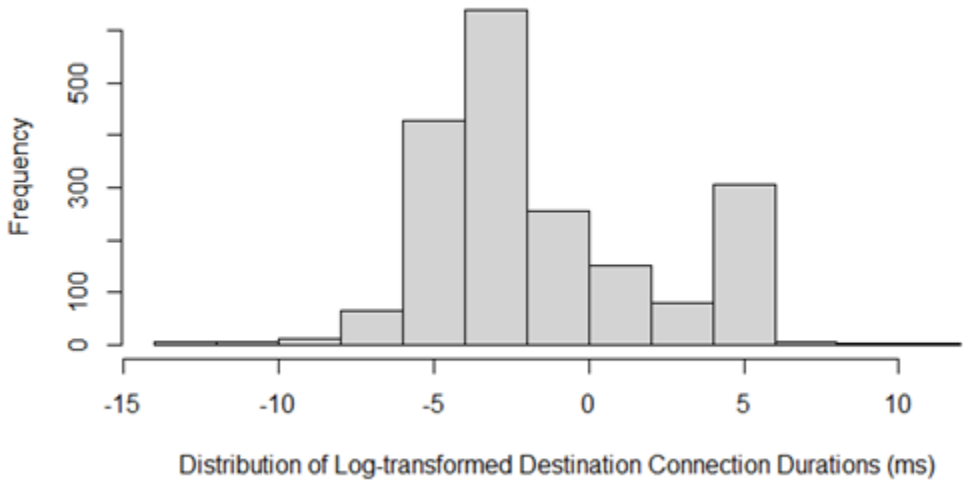


Note. The distribution of network flow destination port addresses

The connection duration (in milliseconds) for each network flow was transformed with a logarithmic scale to provide a more interpretable distribution for analysis. This transformation result is shown in Figure 3, below.

Figure 3

Network Flow Connection Durations (in Log-Transformed Milliseconds) - Honeypot Captured Network Flows for the Curated Benign Class



Note. The distribution of network flow connection durations in log-transformed milliseconds

CHAPTER 4

RESULTS

Research Question 1 Results

As previously stated, the author investigated the first research question, "What do the network flow variables reveal about the distribution of malware classifications in the IoT-23 dataset and the ability to construct non-parametric predictions for the known malware classes?" Specifically, a scientific evaluation of the null hypothesis statement for this research question that using a PERMANOVA methodology was conducted for the null and alternative hypothesis statements:

$H1_0$: The centroids of each of the distinct malware groups' multivariate dependent network flow variables in the IoT-23 dataset are equal.

$H1_a$: There is at least one pair of malware groups with significantly unequal multivariate dependent variable centroids in the network flow dataset.

The author executed an initial PERMANOVA model against the 62,000 records of the IoT-23 network flow observations, using Euclidean distance measurements with variable-dependent relationship ordering, using terms added sequentially (first to last) in the order encountered in the raw IoT-23 network flow dataset, using 200 permutations. This initial test resulted in a p-value for the PERMANOVA model that incorporated terms added sequentially of

0.02985, providing an F value of 0.00017, with which to determine the ratio of explained to unexplained variance.

Utilizing an analysis of the F values for all of the residuals, the research showed 29,998 degrees of freedom with a residual F model score of 0.99983, indicating a strong proportion of explained to unexplained variance across all of the independent variables contained in the model. The model is thus statistically significant with a p-value below the alpha test statistic of 0.05. Thus, the null hypothesis statement that "The centroids of each of the distinct malware groups' multivariate dependent network flow variables in the IoT-23 dataset are equal" is rejected. The author concluded that this dataset sample likely contains patterns that a nonparametric statistical learning algorithm can utilize with which to conduct AI/ML predictive modeling successfully and can be developed with a higher confidence of success.

Research Question 2 Results

RQ2: What are the distinct independent variable cohort relationships between the network traffic flow variables and specific individual IoT malware classes when conducting nominal Internet communications activities upon unrestrained networks?

The null hypothesis statement used for falsifiability testing for this research question was ultimately tested with the following PERMANOVA methodology:

$H2_0$: An independent variable as it contributes to the asymptotic prediction for the dependent variable of malware class (nominal categorical factor variable) is not important, i.e. $\psi_{0,0,j} = 0$ for some j.

$H2_a$: An independent variable as it contributes to the asymptotic prediction for the dependent variable of malware class (nominal categorical factor variable) is important, i.e. $\psi_{0,0,j} > 0$ for some j.

The author produced an initial tree-based (random forest) supervised learning predictive model was for the dependent variable as a baseline effort to explore explainability features in the dataset. This research used a prediction classification model that had 500 trees with a Gini coefficient split rule across 17 independent variables. The resulting cross-validated mean classification error rate was 2.04%, with the resulting predictive output utilized by the next stage in the explainability process. Jethani et al. (2021) defined the FastSHAP algorithm.

The author applied the output from this forest prediction model to the FastSHAP algorithm that incorporated 10 Monte Carlo simulation repetitions. FastSHAP used a gradient descent that converged with contributory cohort explanatory scores for each prediction observation produced by the forest model, a segment of which is in Table 2 below:

Table 2

Shapley Additive Values Provided for the Forest Predictive Model Classifications from FastSHAP (First 10 Observations)

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
1	0.107	0.366	-0.712	-1.73	-0.435	-0.0790	0.0391	0.0138	-0.0194	0.0260
2	-3.97	-0.0468	0.476	0.712	1.63	0.005	-0.073	0.014	0.014	-0.024
3	1.68	2.10	-0.631	-2.38	-1.44	-0.038	0.067	-0.023	0.027	0.03
4	0.470	-2.98	-0.154	4.16	2.06	0.005	0.002	0.038	0.054	0.04
5	-1.99	-0.260	-0.498	2.01	0.806	-0.001	-0.027	-0.027	0.034	0.002
6	-0.309	1.45	-1.29	-2.19	1.46	0.617	0.01	-0.004	-0.003	-0.054
7	0.722	1.54	1.15	-4.57	-0.556	-0.004	0.173	0.03	0.168	0.066
8	-0.842	-1.52	-0.389	3.31	0.0269	0.011	0.035	0.055	-0.007	0.05
9	0.993	2.27	-0.751	3.15	-1.20	-0.056	-0.07	0.026	0.011	-0.04
10	0.345	-1.70	-0.721	-3.99	1.03	0.049	-0.005	-0.006	-0.16	-0.033

Note. A sample of the Shapley additive Values (SHAP explanatory values) from FastSHAP

The research then used a falsifiability assessment test with a PERMANOVA methodology and an adonis2 model for a test of permuted statistical significance. This method

converged the gradient output from the FastSHAP model's Shapley contributory values for each of the 17 independent variables and the predicted classification from the baseline forest model as a new proxy-dependent variable. Using 200 permutations as a generally accepted convenience measure resulted in an alpha test statistic of 0.05 for my Type I statistical error, with terms added sequentially in the PERMANOVA using adonis2. The test resulted in a p-value of 0.001, providing an F value of 0.00035, for determining the ratio of explained to unexplained variance in this model.

An analysis of the pseudo-F values for all of the residuals indicated 30,782 degrees of freedom with a residual F model score of 0.99268, demonstrating a strong proportion of explained to unexplained variance across all of the Shapley score independent variables contained in the model. The model is thus statistically significant with a p-value below the alpha test statistic of 0.05.

The author rejected the null hypothesis statement that "An independent variable as it contributes to the asymptotic prediction for the dependent variable of malware class (nominal categorical factor variable) is not important, i.e. $\psi_{0,0,j} = 0$ for some j". The researcher therefore could not reject the alternative hypothesis statement that "An independent variable as it contributes to the asymptotic prediction for the dependent variable of malware class (nominal categorical factor variable) is important, i.e. $\psi_{0,0,j} > 0$ for some j."

This allowed for the falsifiable conclusion that this dataset contains non-random information with respect to the independent variables containing the Shapley additive values that provide a detailed explanative description of each specific malware predictive classification in the dataset. The Shapley scores for each independent variable that is retained in the explanatory model indicate the deviation that is taking place within this independent variable for each

prediction from the average of the dependent variable classification. For each observation, the total sum of the available Shapley values for each of the available features relates to the total deviation from the mean classification prediction (dependent variable).

Research Question 3 Results

The third research question addressed the question, "What effect does a supervised nonparametric statistical learning model have for determining the presence of malware on human-curated datasets for Raspberry Pi devices using network flow data across an unrestrained network?"

The investigation next used a deep learning multinomial (multiple) classification predictive algorithm to create an asymptotic deep learning multi-layer perceptron with backpropagation features. This employed a deep learning model as a predictive classification model that incorporated subsampled gradient boosted machine predicted model non-parametric output. The author then had the output of this predictive model analyzed by a permuted multiple analysis of variance model to produce pseudo-F statistics with which to generate a p-value to test the relationship between the dependent variable and the independent variables in the model to address the hypothesis test for the third research question:

$H3_0$: All coefficients in the multinomial logistic regression equation will take the value of zero.

$H3_a$: The model currently under consideration is accurate in that it differs significantly from the null coefficient values of zero; it gives significantly better than random chance predictive strength relative to the null hypothesis.

Using a Chi-square Goodness of Fit Test, it was determined that a randomly generated n=62,000 IoT-23 flow observations would provide sufficient statistical power for small effect size measurement using the dependent variable categorical factors present.

This allowed for the production of a minimal sample size estimate using the Chi-square model parameters of a significance level (α) for the Type I statistical error at 0.01, for measuring a "small" effect size (w) of 0.03, a statistical power of 0.99, for 23 distinct factor levels within the dependent variable. The sample size estimate produced by this Goodness of Fit test resulted in an n=54,741 that gained the statistical power of 0.990001 for training our deep learning (multi-layer) feed-forward predictive model for RQ3. To enhance the conservative nature of this approach, an n=75,000 was selected randomly from the total population using class balancing with the dplyr package in the R statistical computing environment, as discussed in Broatch et al. (2019).

The initial multi-layer perceptron deep learning neural network incorporated seventeen independent variables arranged in three neuron layers arranged in a 40,40,40 neuron distribution across three "deep learning" hidden layers. Each of these hidden layers represents the non-linear transformation space where the input data processes through a Tanh activation function. In this way, a non-parametric statistical learning framework took place before the computed results passed to the final output layer of the neural network.

The exact parameter configurations including the neural network weights included allocating 50 gigabytes (GB) of random-access memory space for the deep learning neural network model using the h2o library package, with unlimited available multiple processor threads allocated through the use of the R statistical computing environment. Qualitative character data was first converted to multi-level factor data in the independent variables and the

dependent variable within R (nominal data types), and then those nominal values were re-coded to be ordinal numeric using the R as numeric function, which re-coded the variables to ordinal data types based alphabetical order for each factor level.

This dummy variable re-coding was necessary to support the use of the PERMANOVA methodology using the adonis2 function from the Vegan library package in the next phase of investigation for this Research Question. Vegan package functions require numerical data to function for both the IVs and the DVs, therefore recoding multi-level factor data for use in those functions is a requirement. Therefore, the author recoded the nominal multi-level factor data before processing by the deep learning neural network within the h2o library package model to keep the data in complete alignment throughout each phase of the analysis for this Research Question.

The parameter configurations for the h2o neural network predictive classification model consisted of a multinomial distribution (classification) setting, three hidden layers of 40 neurons each, 17 independent (predictor) variables, 10 epochs (how many times the dataset should be iterated), using a Tanh activation function with the balancing classes parameter set to TRUE inside the neural network model configuration. This methodology enabled an adaptive learning rate feature as TRUE, allowing the use of h2o's ADADELTA algorithm as discussed in Zeiler (2012). The ADADELTA algorithm functions within the constraints of the h2o deep learning neural network to pool the benefits of both the learning rate annealing with momentum training, with the goal of avoiding inefficient, slow model convergence.

In addition, a random seed value provided a starting point for the model convergence (seed value of 12345). Variable importance calculations were applied as an additional model output (key response indicators or KRIs for the predictor variables). A defined stopping tolerance

was set to 0.01. A train/test split of the 75,000 statistically balanced training data was defined as being a ratio of 0.70 train and 0.30 for testing based on random selections within the n=75,000 observation dataset.

The train/test validation split ratio within the subsampled (and statistically balanced) dataset that was provided to the model was specified so that it could assess the algorithmic ability to conduct nonparametric statistical learning (against the test partition), rather than assessing how well the model "memorized" the training data partition which was used to train and converge the model. This provides an external validation method to assess the model performance using data that was not part of the training split, while curated human-labeled outcomes data was available to validate against the predictions from the model after training.

The researcher primarily measured the performance of this fully converged and trained deep learning model using this test partition split, with what the h2o library package refers to as the "validation set metrics." The validation set metrics for the RQ3 Deep Learning Classification Prediction Model are shown in Table 3, below:

Table 3

H2o Algorithm Deep Learning Predictive Classification Performance Metrics for the Test Partition of the Training Data

H2OMultinomialMetrics:	Deep Learning
Reported on validation data	
Metrics reported on temporary validation frame with 99988 samples	
Validation Set Metrics:	
MSE	0.03755783
RMSE	0.1937984
Logloss	0.1752766
Mean Per-Class Error	0.09105377

Note. Predictive model performance validation metrics produced by the RQ3 model

The model supplied the following Top-10 Hit Ratio performance metrics, with the h2o package defining the hit ratios as the number of times that a correct prediction was made in ratio to the number of total predictions made for a given epoch (iteration through the dataset), as shown in Table 4, below:

Table 4

Top 10 (Training Epochs) Hit Ratio Performance Metrics

k	Hit Ratio
1	0.965596
2	0.992669
3	0.999090
4	0.999330
5	0.999760
6	0.999890
7	0.999960
8	0.999960
9	0.999970
10	0.999970

Note. Steadily-improved model training performance indicated across successive epochs for the RQ3 predictive model

The hit ratio performance metrics indicate the percentage that a correct prediction was made on the test portion of the training dataset that was withheld from training the model and used to externally validate model performance. As the epochs increase, the deep learning neural network steadily improves (reduced Type I error) as indicated by the hit ratio.

The deep learning prediction classification model produced for Research Question 3 can thus be interpreted as being highly effective, and demonstrably improving its learning over time (epochs) based on the MSE or Mean Square of the Error of 3.75%, and a Mean Per-Class Error

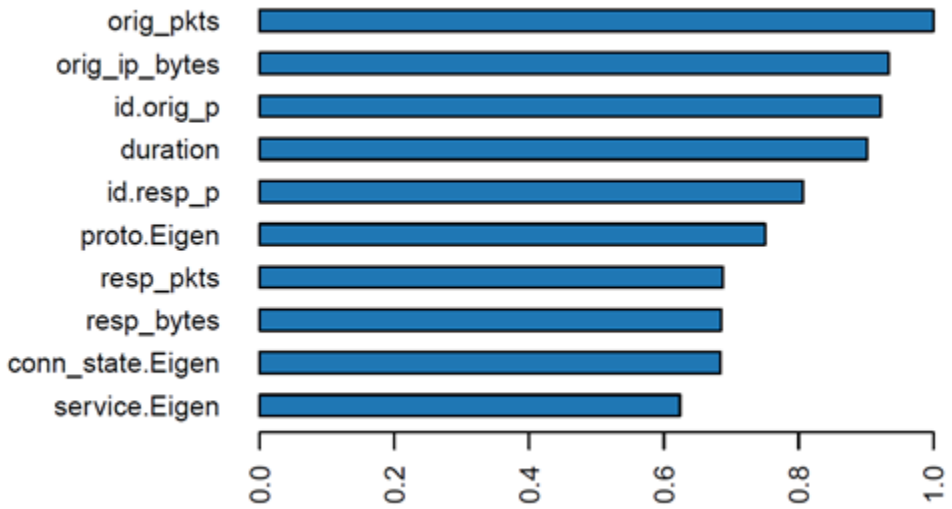
rate of 9.1% across all 23 dependent variable factor classes present in the data. Likewise, the hit ratio shown in Table 4 above indicates a high degree of performance accuracy as well as a steady optimization improvement during the stochastic gradient descent process, as model performance for each iteration epoch (k) improves marginally as the algorithm repeatedly traverses through the available data, determining the optimal gradient loss function.

The deep learning predictive model is thus effectively accurate at predicting the known malware classes, including the benign classes identified in the curated dataset regardless of class observation frequency within the population of the dataset. Further, the author could state that the validation portion of the dataset indicates that the model does an extremely robust job of learning from the nonparametric statistical patterns that are present in the training dataset, rather than simply memorizing the most likely majority classes present in the training data.

This research then used the deep learning prediction model to produce a variable importance plot, using the “varimp” function for the h2o deep learning library package against the trained neural network prediction model. The variable importance plot produced by this algorithm utilizes an approach pioneered by Gedeon (1997), based on the methodology of feature weights connected from the input layer to the first two hidden layers within the deep learning neural network. The resulting chart plot (Figure 4, below) shows the relative weights for the selected predictor (independent) variables using a standardized scale, to provide a sense of the relative ranked importance as selected by the deep learning predictive model.

Figure 4

Variable Importance: Deep Learning Model



Note. The top 10 key response indicator variables showing relative importance from the RQ3 classification model

The variable importance plot produced by the h2o algorithm deep learning neural network predictive model indicated the relative importance on a standardized scale of the independent variables utilized by the successfully converged neural network, based on a scaled importance score as shown in the details contained in Table 5. The variable importance plot represents the overall set of features ("key response indicators" or KRIs) in ranked order that as utilized by the successfully converged deep learning neural network and represents the independent variables likely to have the most significant non-parametric statistical relationship to the dependent variable in the available dataset. A more detailed tabular breakdown of these 10 Key Response Indicator (KRI) variables is in Table 5 (below) provides the percentage

importance of each variable to the overall model, as well as the scaled importance for each of these independent variables.

Table 5

Detailed Table of KRI Independent Variables for RQ3

	Variable	Scaled_importance	Percentage
1	orig_pkts	1	0.104007661
2	orig_ip_bytes	0.933485389	0.097089632
3	id.orig_p	0.922122955	0.095907852
4	Duration	0.901571691	0.093770363
5	id.resp_p	0.806919217	0.08392578
6	proto.Eigen	0.750077069	0.078013762
7	resp_pkts	0.687973499	0.071554515
8	resp_bytes	0.6850003	0.071245279
9	conn_state.Eigen	0.683816731	0.071122179
10	service.Eigen	0.623869777	0.064887236

Note. A detailed tabular breakdown of relative scaled importance and percentage importance for the KRIs produced by the RQ3 model

The most important contributory independent variables for the RQ3 prediction model are shown in ranked order, using percentage scores and scaled importance values. Using those "top 10" KRI variables provided by the variable importance plot, a PERMANOVA model was then produced using the malware class ("detailed label") dependent variable and the 10 KRI variables provided by the deep learning model. The intent of the use of this PERMANOVA model is to create a permuted test of statistical significance of the findings of the deep learning model.

To further reinforce the effectiveness of the predictive classification deep learning model developed using the h2o library package and to support the assertion of falsifiability in addition to simply showing prediction accuracies, the author then developed a PERMANOVA model. This permutedly analyzed the available output from the deep learning prediction classification

model produced for Research Question 3. The results of this PERMANOVA model, conducted via the "adonis2" function in the Vegan library package for the R statistical computing environment using calculated non-Euclidean Bray-Curtis distances, are in Table 6. Table 6 below provides the coefficients and residuals produced by an Adonis2 permuted MANOVA model from the Vegan library package, utilizing Euclidean distance separations and 200 permutations.

Table 6

PERMANOVA Adonis2 Non-Euclidean Bray-Curtis Distances

	Degrees of Freedom	Sum of Squares	R2	F	Pr(>F)
Model	10	94523	0.14305	667.55	0.004975
Residual	39989	566233	0.85695		
Total	39999	660756	1.00000		

Note. PERMANOVA model showing permuted residuals and coefficients based on an analysis of RQ3 model output

Against an alpha test statistic of 0.05 for the Type I statistical errors, the dependent variable is permutedly statistically significant with a p-value of 0.004975 against an alpha test statistic (Type I error) of 0.05.

The PERMANOVA (permuted multiple analysis of variance) model allowed me to conduct a test of non-parametric statistical falsifiability of the relationship of the deep learning model predictive output generated using the statistically balanced classes in a randomly selected distribution based on n=75,000 in keeping with the sample size requirements from the previously produced Chi-square Goodness of Fit Test.

Using 200 permutations as a generally accepted convenience measure as well as for allowing for limitations in available computing resources, produced a stated alpha test statistic for our Type I statistical error at 0.05. Terms were added sequentially to the PERMANOVA

model using the “adonis2” function from the Vegan library package, resulting in a permuted p-value of 0.004975, and an F value of 667.55. The F value provides the ratio of explained to unexplained variance in the model. This provides evidence that the variation between sample means is high enough relative to the variation within the samples to allow me to reject the null hypothesis, “All coefficients in the multinomial logistic regression equation will take the value of zero.”

The research therefore rejected the null hypothesis statement and failed to reject the alternative hypothesis $H3_a$, “The model currently under consideration is accurate in that it differs significant from the null coefficient values of zero; it gives significantly better than random chance predictive strength relative to the null hypothesis.”

Research Question 4 Results

The fourth research question addressed the use of self-supervised, unsupervised deep learning models and their ability to determine the presence of human-curated malware classes without having pre-training data for the labeled class identities in a response variable.

The hypothesis statements for this fourth and final research question were:

$H4_0$: There is no statistically significant relationship between the predictor variables and the response variable in the anomaly-driven response variable groups.

H4a: There is a statistically significant relationship between the predictor variables and the anomaly-driven response variable groups.

The methodology employed to investigate this research question primarily involved the concept of self-supervised, unsupervised autoencoding deep learning models. The purpose of the models employed for this research question was to effectively determine the presence of human-curated malware classes without having to be pre-trained on the labeled class identities. The

employed unsupervised deep learning models therefore constitute the final component of a defense-in-depth strategy for defensive cyber analytics using artificial intelligence that incorporates nonparametric, permuted falsifiability testing at each layer. Ultimately, this final set of models for Research Question Four demonstrated the ability of the ensemble supervised and unsupervised AI/ML systems to identify novel malware classes without pre-training with human-curated examples of malicious network intrusions.

To mitigate the loss of granularity in the model from the available data by not utilizing the human-curated response variable in this neural network, the response variable from the available curated data was recoded to a binary value of 0 or 1. These values represented "malicious" and "benign" in the available data, and only for purposes of performance validation of the unsupervised prediction model for anomaly detection. The author excluded the response variable itself from the dataset employed by the autoencoding neural network for this research question.

This response variable recoding addressed a fundamental aim of the fourth research question, with regard to detecting the presence of malware classes but not necessarily being able to enable the identification of distinct specific entities. The goal of this research question was to allow for the presence of a "backstop" methodology, suitable for the detection of all malware types, known and novel. As such, it was not necessary to have the unsupervised anomaly detection algorithm to be able to discern specific sub-types of malware, as this was addressed in previous research questions in the paper. Being able to identify on a Boolean basis whether a malware attack is present or not without using human-curated labels and with high probabilistic confidence for any given network flow observation would address this final research question.

The author next ingested a randomly selected subsample dataset of n=146,684 to provide an extremely conservative sample size in keeping with the previous Chi-square Goodness of Fit minimum sample size (n=75,000) for 23 distinct classes for a small effect size with sufficient statistical power of 0.99. This was performed to account for finite computing resources. This sample used the recoded response variable into the h2o neural network library package in a separate computing cluster environment consisting of 50 gigabytes of RAM and 32 available CPU cores. The four qualitative multi-level factor variables were configured in the h2o environment as being factor-type predictor variables.

The research then undertook a grid search parameter optimization process to determine the optimal configuration of this autoencoding neural net. The resulting optimal deep learning self-supervised, unsupervised learning autoencoder model contained the parameter space of 16 predictors, no response variable, with two hidden layers of 13 neurons each and using 5 epochs (dataset iterations). The use of multiple hidden layers within the autoencoding neural network defined this model as a “deep learning” autoencoder.

Assessing the performance of the autoencoder anomaly scores against the external human-labeled response variable that was withheld from the autoencoder model, the author assessed that the model performed at over 91% accuracy in detecting malicious versus benign network flow types across all classes of known malware. In order to assess with falsifiable certainty that this model was permutedly statistically significant, Anderson's “PERMDISP2” (permuted beta dispersion) procedure was utilized for the analysis of multivariate homogeneity of group dispersions (variances), from the Vegan library package in R. The Beta dispersion test is a multivariate analogue of Levene's test for homogeneity of variances, using non-Euclidean distances between objects and group centers. This methodology produced the distinct group

centers (i.e., centroids or medians) by reducing the original distances to principal coordinates from the original multi-dimensional feature space. In keeping with supporting practices, this procedure was applied latterly as a means of assessing beta diversity between the two distinct groups in the predicted anomaly response.

As a requirement of addressing limited (50 GB RAM) computational environment resources for a permuted statistical falsifiability test, the author randomly sampled 40,000 observations from the original anomaly prediction dataset of 146,684 observations. This test calculated the beta dispersion using the "vegdist" (Vegan distances) which produced Bray-Curtis non-Euclidean distance separation values for the centroids of both anomaly response types.

Table 7

Beta Dispersion Table for the Homogeneity of Multivariate Dispersions

No. of positive Eigenvalues:	1887							
No. of negative Eigenvalues:	3560							
Average distance to median:								
Benign malicious	0.3640	0.2801						
Eigenvalues for PCoA axes:	PCoA1: 2762.98	PCoA2: 1920.99	PCoA3: 1436.38	PCoA4: 507.96	PCoA5: 273.90	PCoA6: 159.07	PCoA7: 113.66	PCoA8: 73.54

Note. The principal component axes and their respective Eigenvalues as determined for the Bray-Curtis centroid distances between the "benign" and "malicious" composite groups

Table 7 above provides Homogeneity of Multivariate Dispersions to indicate the the different Bray-Curtis centroid distances to the median between the two distinct groups (Benign and Malicious). This was calculated by the deep learning autoencoder model, with the Eigenvalues used across the 8 principal component axes. Finally, a permutation test was

conducted for F using the Vegan library package “permutest” (permuted test) to non-parametrically, statistically test for homogeneity of multivariate dispersions between groups. Using 99 permutations as a generally accepted convenience due to limited computational resources for a very computational resource-intensive process, using the calculated Bray-Curtis distances as the response (dependent) variable in this model, resulted in the model coefficients and residuals as shown in Table 8, below:

Table 8

Bray-Curtis Model Coefficients and Residuals

Response:	Distances					
	Df	SumSq	MeanSq	F	N.Perm	Pr(>F)
Groups	1	70.3	70.297	1160.5	99	0.01
Residuals	39998	2422.9	0.061			

Note. Residuals provided by the Bray-Custis distance model to determine the statistical significance of the distance between the two composite groups

Table 8 provides the residuals and coefficient values provided by a permuted test using Bray-Curtis non-Euclidean distances of the multivariate dispersions between the two groups in our response variable, achieved using 99 permutations against an alpha test (Type I error) statistic of 0.05.

The result of this permuted test against a stated alpha test statistic of 0.05 was a p-value of 0.01 that was statistically significant against the null hypothesis statement that “There is no statistically significant relationship between the predictor variables and the response variable in the anomaly-driven response variable groups.” Therefore, the research rejected the null hypothesis statement, with a failure to reject the alternative hypothesis statement, “There is a

statistically significant relationship between the predictor variables and the anomaly-driven response variable groups."

Discussion

Discussion of Research Question 1 Findings

As a means of visualization for Research Question 1, the research analyzed the transformed principal component centroid deviations for each treatment group (multi-level factor variable dependent variable) using a Homogeneity of Multivariate Dispersions test. This test utilized methods from Oksanen et al. (2013), specifically a permuted multivariate variation of Levene's test for homogeneity of variances. Due to the multivariate permuted nature of this test, the methodology is extremely computationally intensive and therefore required the use of randomized balanced class statistical subsampling from the larger, original labeled dataset to conduct the beta dispersion analysis. For this analysis, a randomly selected and statistically balanced sample size of n = 500 observations for each curated labeled class group in the dependent variable was made.

The result of this further permuted test demonstrated a significant multivariate dispersion that was measurable for the malware and benign class groups within this subsampling, as shown in Table 9 (below):

Table 9

Homogeneity of Multivariate Dispersions

No. of positive Eigenvalues:	652					
No. of negative Eigenvalues:	1385					
Average distance to centroid median:	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
	3.652e-01	1.153e-01	9.759e-02	9.281e-02	1.408e-01	1.298e-01
	Class 7	Class 8	Class 9	Class 10	Class 11	Class 12
	7.369e-02	5.806e-02	7.403e-02	3.555e-01	1.851e-01	1.869e-01
	Class 13	Class 14	Class 15			
	1.012e-01	2.174e-01	6.436e-05			
Eigenvalues for PCoA axes (showing 8 of 2037 Eigenvalues):	PCoA1: 695.63	PCoA2: 343.28	PCoA3: 132.23	PCoA4: 58.44	PCoA5: 55.14	PCoA6: 33.34
	PCoA7: 21.42	PCoA8: 20.34				

Note. The non-Euclidean centroid distances between 15 malware classes, and the 8 principal component axes Eigenvalues used by this homogeneity of multivariate dispersions model

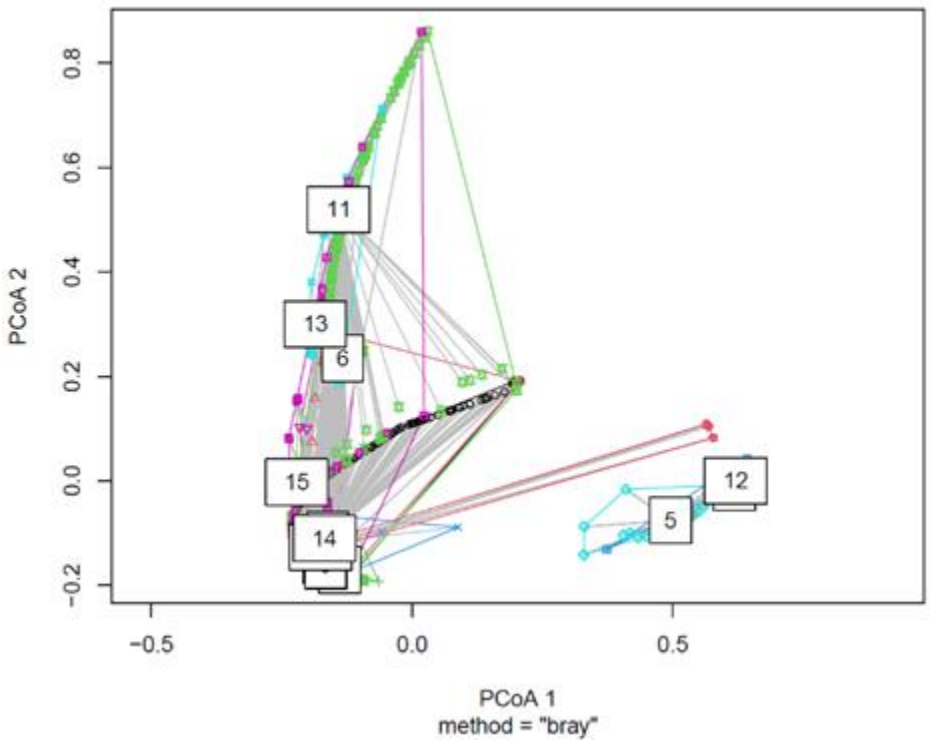
Table 9 provides measurable multivariate beta dispersion based on permuted methodologies developed by Oksanen et al. (2013), for n = 500 balanced and randomly subsampled observations for each curated dependent variable class factor in the original labeled network traffic flows. This is an analogue of Levene's test of homogeneity of variances, utilizing non-Euclidean distances between group centroids (media) using principal component coordinate transformations of multivariate data.

As shown in Table 6, the resulting values from the multivariate homogeneity of group dispersions, known as a beta dispersion test, utilizes Anderson's (2006) non-Euclidean (Bray's method) as a permuted measure of beta diversity between groups. This procedure employs a

principal component coordinate transformation of the independent variables, utilizing Eigenvalue measures. This provides the group of samples with a measure of variance between groups based on the transformed principal component group centroids of non-Euclidean spatial medians within a multivariate space. Figure 5 (below) depicts a principal coordinate analysis using two principal component dimensions and Figure 6 (below) displays skew and distribution between the distinct malware class groups.

Figure 5

Human-curated Observations Beta Disperion of Balanced DV Classes

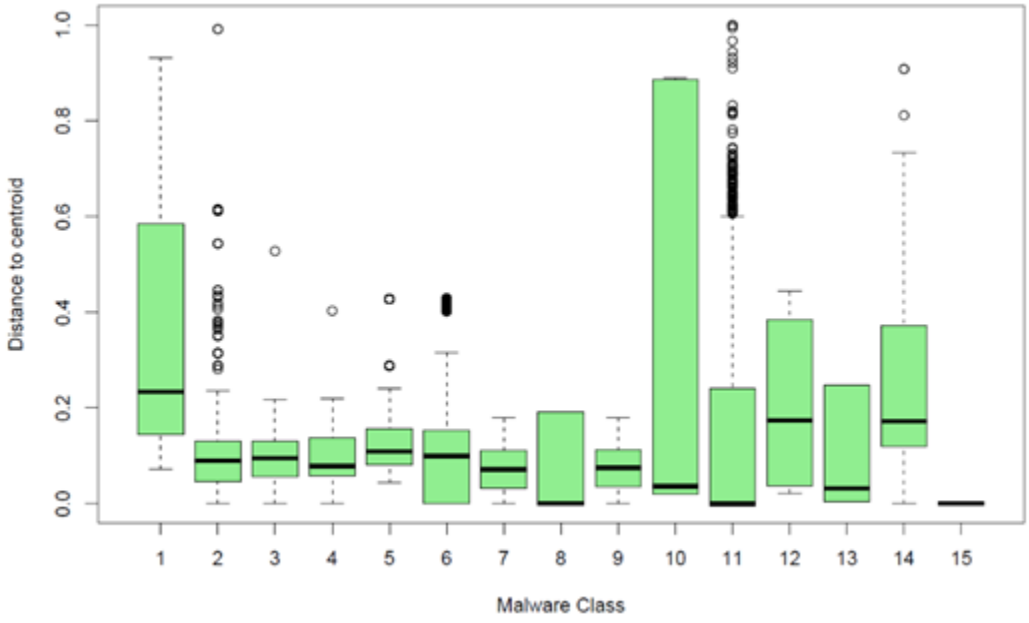


Note. A chart plot showing the non-Euclidean distances across two transformed principal component axes of the human-curated malware class observations

Figure 5 provides 2-dimensional principal coordinate analysis plot showing the transformed non-Euclidean multivariate distances between measurable malware class groups in the dependent variable, visually demonstrating compactness within and non-Euclidean separations between groups. The Bray-Curtis Dissimilarity Beta Diversity method measures variances for between-groups composition medians across multiple independent variables for a given sampling population, employed using randomized class balancing for unequal group sizes (n=500).

Figure 6

Boxplot of Raw Data Malware Classes Distance to Centroids



Note. A boxplot showing the inter-quartile ranges of the distance to centroids for the transformed dependent variable for the most frequent malware classes

Figure 6 provides a boxplot visualization of the Bray-Curtis non-Euclidean centroid distances for each of the 15 measurable (sufficient variance detected) in the n=500 balanced class dependent variable. A distinct measurable set of variance markers is visible in this illustration of skewness and distribution within the sampled group population, supporting the conclusion to reject the stated null hypothesis in favor of the alternative hypothesis.

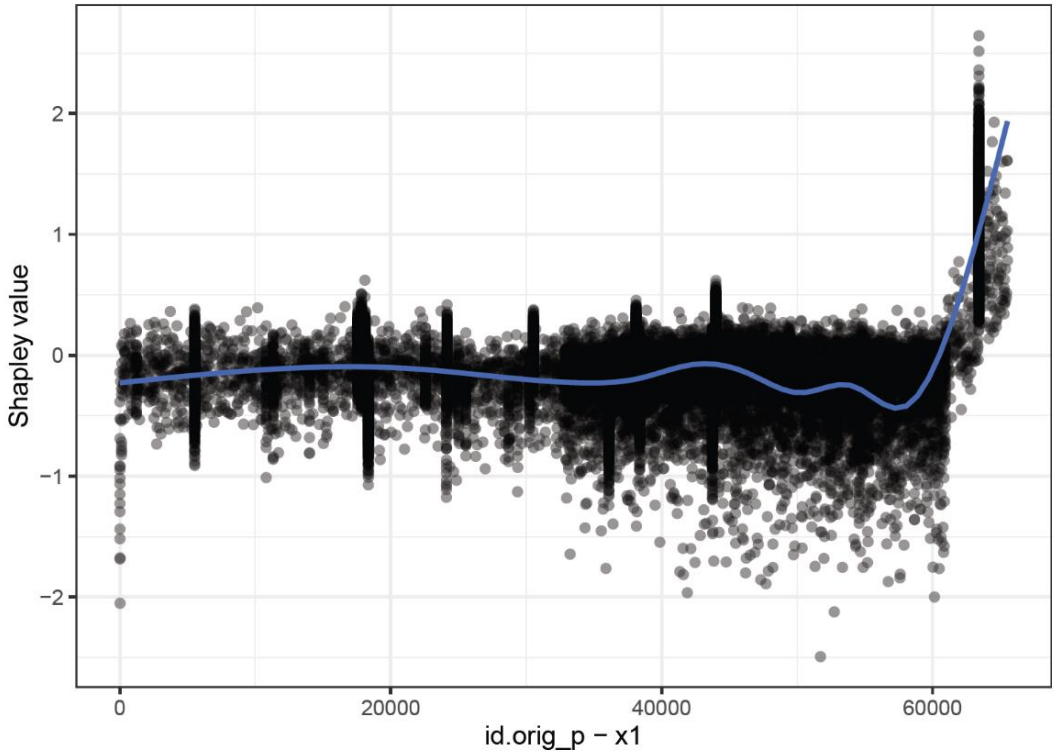
The visualizations contained in both the box plot in Figure 11 and the Beta Dispersion of Malware Class Groups shown in Figure 1 greatly reinforce the permuted MANOVA non-parametric statistical findings that the null hypothesis should be rejected, in favor of the alternative hypothesis. That alternative hypothesis for Research Question One stated, “There is at least one pair of malware groups with significantly unequal multivariate dependent variable centroids in the network flow dataset.”

Discussion of Research Question 2 Findings

In further discussing the findings from Research Question 2, a visualization (Figure 7, below) provides a scatterplot distribution of the x1 independent variable Shapley explanative values from the explanatory model of the additive explanations provided by the “FastSHAP” Shapley scores algorithm. Variable x1 in this “FastSHAP” model relates to the IoT-23 dataset, while variable id.orig_p (origination port) in the network flow dataset shows deviation within that specific IV for each prediction.

Figure 7

Shapley Values for id.orig_p with a Local (Univariate) Polynomial Regression



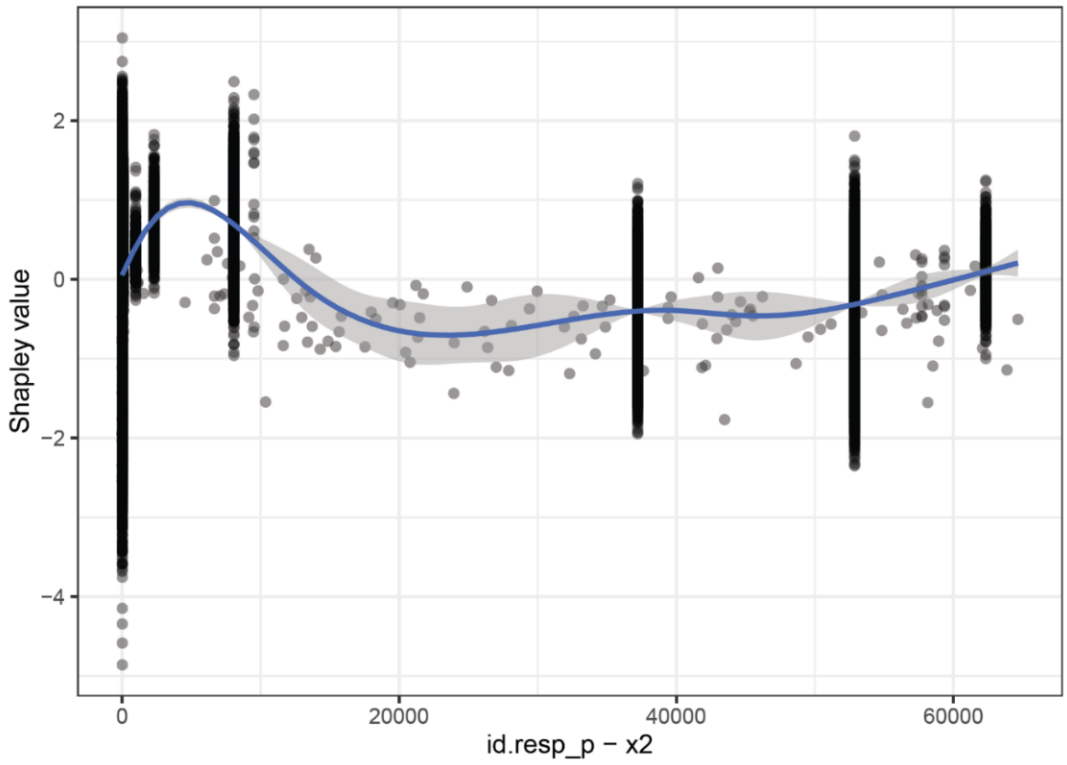
Note. SHAP explanatory values for the id.orig_p variable with a splined Local (Univariate) Polynomial Regression or LOWESS projection in blue

Figure 7 provides Shapley values for id.orig_p (Origination Port) with a LOWESS (locally weighted scatterplot smoothing) projection line in blue, also known as a local (univariate) polynomial regression. The individual Shapley observation values provide the deviation within the independent variable from the average of the dependent variable per classification prediction. For each observation, the sum of the available Shapley values for each of the available features relates to the total deviation from the mean classification prediction.

Figure 8 below likewise demonstrates the x2 variable from the “FastSHAP” library package explanatory model, indicating the deviance of each observation per classification prediction for the id.resp_p IoT-23 network flow dataset variable (response port).

Figure 8

The id.resp_p (Response Port) Shapley Additive Value Scores Scatterplot



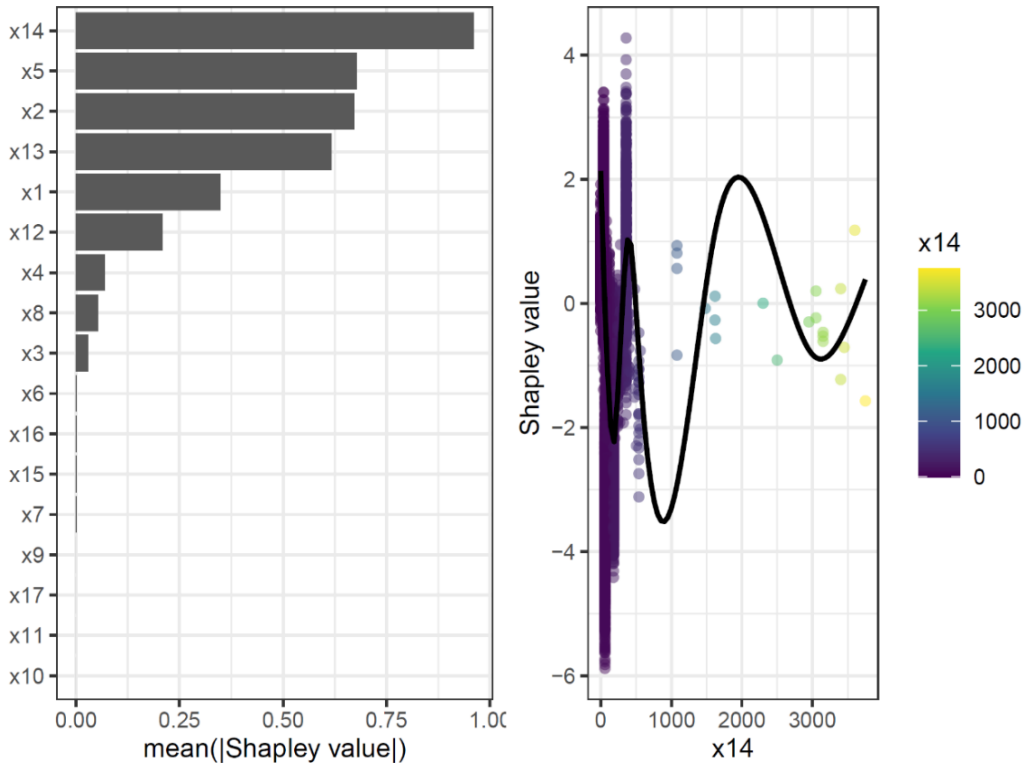
Note. SHAP explanatory values for the id.resp_p variable with a splined Local (Univariate) Polynomial Regression or LOWESS projection in blue

Figure 8 provides the id.resp_p (response port) Shapley additive value scores (deviance per prediction observation) in a scatterplot distribution for a LOWESS local polynomial regression trend in blue, showing a deviance signature for each prediction classification.

Figure 9 provides an overview of all of the Shapley features in order to significance (in terms of mean deviance value) on the left-hand side. Figure 6 (below, right) also provides a detailed scatterplot matrix with a splined LOWESS univariate trend for the behavior of variable x14, the independent variable feature that contained the largest amount of variance in the dataset. Variable x14 translates to the Orig_IP_bytes (origination IP address byte count per flow) in the IoT-23 network flow dataset.

Figure 9

Mean Shapley Values (Deviance Prediction Mean) and Independent Variable X14 Scatterplot Matrix



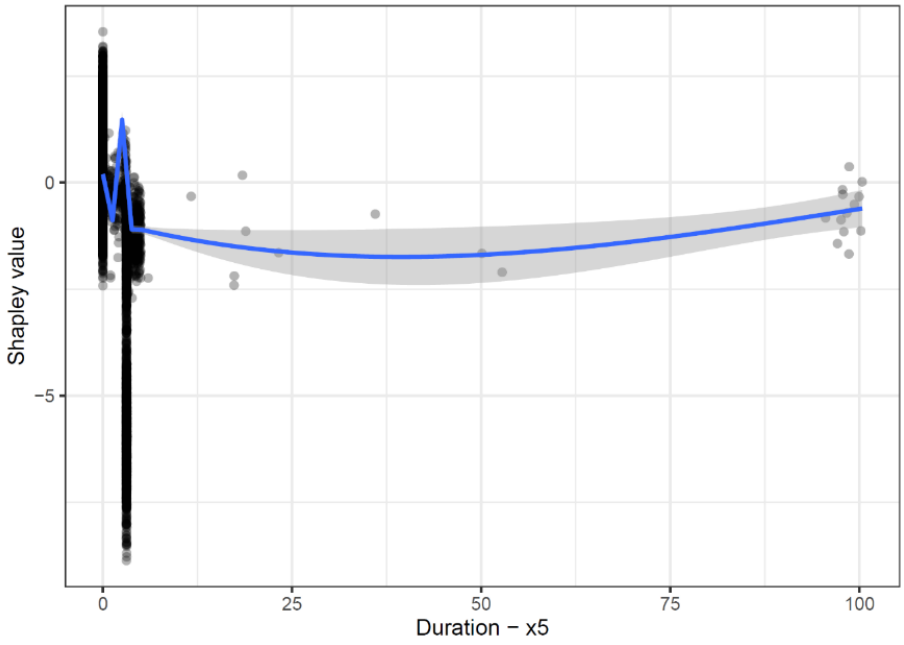
Note. Mean SHAP values for all available key response indicator features, with a color-coded scatterplot of variable x14 (origination IP address byte count per flow), the most important contributing predictor variable

Figure 9 provides mean Shapley values (deviance prediction mean) for all available features and a color-coded scatterplot matrix of the independent (predictor) variable x14, the most significant Shapley feature in terms of mean score (mean deviance per prediction). X14 relates to the IoT-23 dataset independent variable Orig_IP_bytes (origination IP address byte count per flow).

Figure 10 provides a scatterplot matrix of Shapley value variable x5 that corresponds to the Duration predictor variable in the IoT-23 network flow dataset. Banding is visible in this and the other Shapley feature variable scatterplots that correspond with the majority of the malware event classes recorded in the dataset.

Figure 10

x5 Shapely Feature Variable and Duration Independent Variable in the IoT-23 Network Flow Dataset



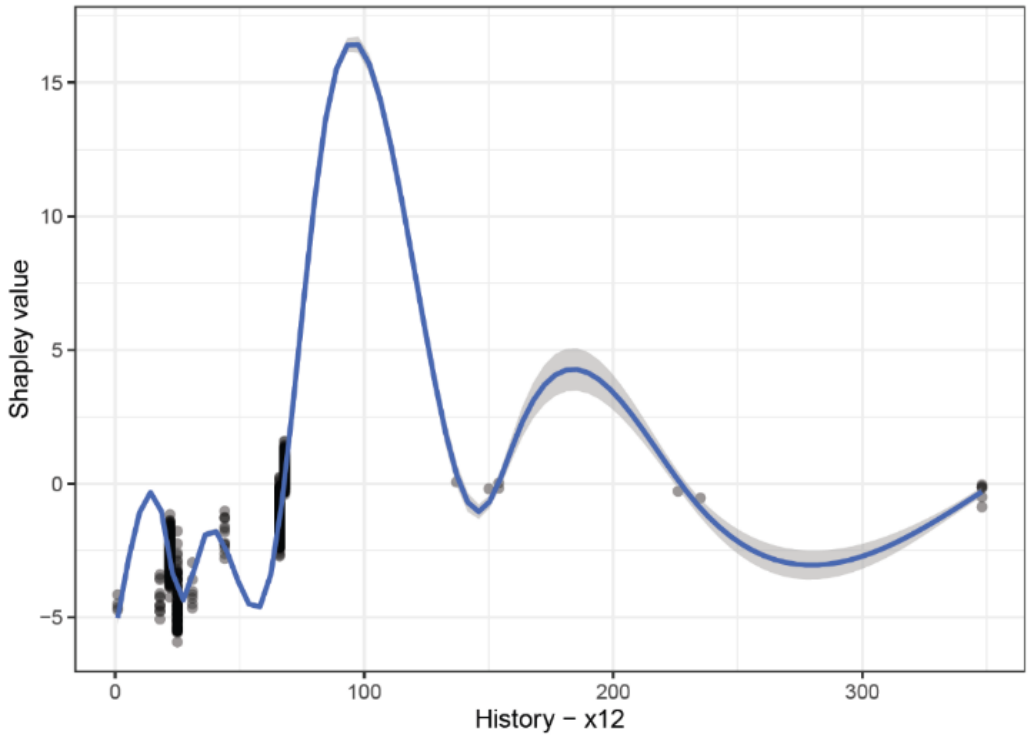
Note. SHAP explanatory values for the x5 (Duration) variable with a splined Local (Univariate) Polynomial Regression or LOWESS projection in blue

In Figure 10, the x5 Shapley feature variable corresponds with the Duration independent variable in the IoT-23 network flow dataset. The original Duration variable indicated connection times in milliseconds during the network flow session, and in this scatterplot, the Shapley values indicate variance for each prediction observation.

Figure 11 illustrates a scatterplot of Shapley value distributions showing the deviance distribution per malware class prediction for the x12 variable, representing History in the IoT-23 network flow dataset, as shown below:

Figure 11

Shapely Value Distributions Deviance Distribution per x12 Malware Scatterplot



Note. SHAP explanatory values for the x12 (History) variable with a splined Local (Univariate) Polynomial Regression or LOWESS projection in blue

Discussion of Research Question 3 Findings

To provide further analysis of the results from Research Question 3, it was concluded that this dataset contains non-random information with respect to the set of independent variables as a whole for the dataset in our population. With respect to their predictive relationship for the 23 specific classes of known malware types, our suites of independent variables within our population contain linear and non-linear predictive strength indicating a causality relationship to our dependent variable.

The deep learning prediction model also made use of Eigen factor categorical encoding, with respect to the multi-level (qualitative) factor variables present in the data (nominal type statistical information). Those qualitative, nominal factor variables included the proto (protocol), conn_state (connection state), and service variables. Internal to the h2o deep learning multi-layer perceptron neural network algorithm, those nominal type variables were encoded using Eigen factor relationships to the other independent variables in the dataset, rather than utilizing traditional "one hot encoding" (e.g., N+1 new columns for categorical features with N levels). Similar methods to one hot encoding such as binary encoding (up to a maximum of 32 columns per categorical feature) might also typically be employed in these scenarios to encode these qualitative values into quantitative value representations. However, by utilization of Eigen encoding, these multi-factor variables of *k* columns per categorical feature were transformed by projecting an otherwise one-hot-encoded matrix onto *k*-dimensional Eigen spaces only for each independent variable that the categorical variable has a relationship with, as determined by the deep learning predictive model. This has the potential to efficiently encode the qualitative data into discrete quantitative values internally within the model but does so far more efficiently in

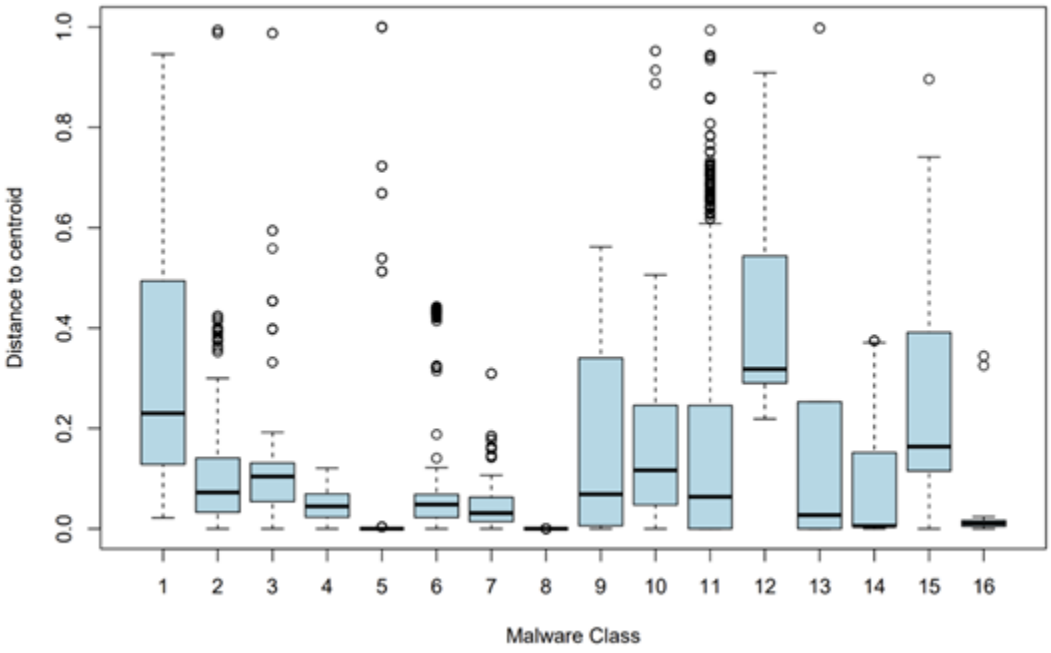
terms of memory utilization, allowing the model to converge much more rapidly while enabling more efficient use of the neuron weights and layers available to it.

The result is a transformation of the original qualitative nominal independent variables into Eigen-transformed quantitative values, with the variables renamed accordingly as shown in Figure 9 (e.g., “proto.Eigen”, versus the original “proto”).

To help visualize the patterns within our complex dataset, a box plot of the distinct malware factor classes present within the dependent variable based on the distance to the centroids for the transformed set of independent variables in the dataset as shown in Figure 12 below:

Figure 12

Boxplot of Predictive Model Malware Classes Distance to Centroids



Note. A boxplot showing the inter-quartile ranges of the distance to centroids for the transformed set of independent variables for the most frequent malware classes

Figure 12 is a box plot showing the inter-quartile ranges based on the distance to the centroids for the transformed set of independent variables for the most commonly occurring malware classes.

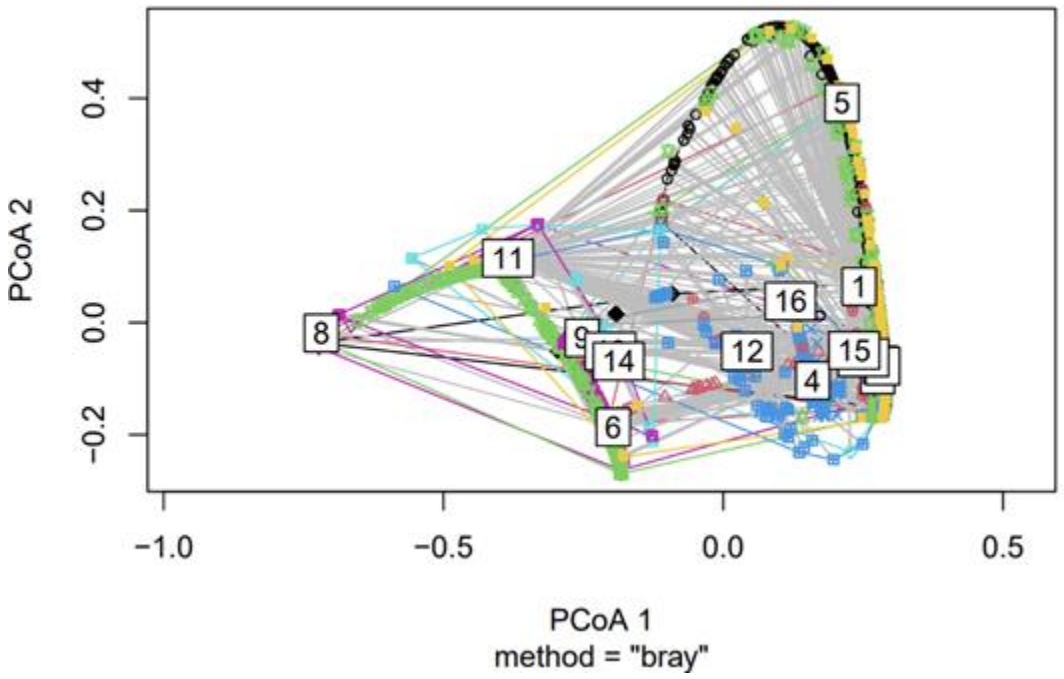
The box plot from Figure 8 shows significant variance across the inter-quartile ranges, medians, and outlier relationships of the transformed predictor variables as determined by the principal components of this dataset. This demonstrates the strong and statistically significant predictive relationship proven by the PERMANOVA model for this research question.

Likewise, a Beta dispersion model utilized this same PERMANOVA model, using calculated Bray-Curstis (non-Euclidean) distances for each of the malware classes indicated in the predictive model forecast data.

These Bray-Curtis (non-Euclidean) distance calculations established by the PERMANOVA model allowed us to determine a standardized multivariate dispersion between groups as part of a test of homogeneity between groups across the principal component axes. This resulted in a distance separation plot in Figure 10 below, allowing us to visualize across the transformed principal component dimensions of the available predictor variables what the multivariate dispersion between groups is within this vector space. The resulting distances shown in Figure 13 provide the Eigenvalues of the multivariate centroids for each distinct malware class in the predictive model population. This visualization provides additional evidence for the distinctness (non-homogeneity) of each distinct malware class, reinforcing the findings of predictability of each class factor by the deep learning predictive model.

Figure 13

Predictive Model Beta Dispersion of Balanced DV Classes



Note. A chart plot showing the non-Euclidean distances across two transformed principal component axes of the predictive model malware class forecast

Figure 13 shows the beta dispersion of the centroids for each balanced malware class for Research Question 3 in the DV from the subsampled predictive model forecast output. These distances utilize the deep learning neural network predictions analyzed by the PERMANOVA model across the principal component dimensions of the predictor variables. The Beta Dispersion chart indicates that the various malware classes in the dataset are clearly separated across the principal component space, indicating that the malware classes are distinct (non-homogenous) with significant non-Euclidean distances between each group, improving the likelihood of predictability for each group (malware class).

Discussion of Research Question 4 Findings

Lastly, the author provide further discussion and insight for Research Question 4. This began by conducting a pairwise comparison between the two summary groups of benign and malicious, providing a pairwise comparison permuted p-value between the two groups as shown in Table 9. This test was conducted by using an analytic permuted t-distribution, producing permuted p-values obtained from 99 permutations. The parametric assumptions of the t-test are satisfied by the extremely low permuted p-values obtained for both between-group comparisons, shown in Table 10, below:

Table 10

Beta Dispersion (Vegan library package) Pairwise Comparison

Observed p-value below diagonal	Permuted p-value above diagonal
Benign	Malicious
0.01	9.1268e-251

Note. A pairwise comparison of the beta dispersion, comparing the centroid distances of the mean Bray-Curtis distances between the two composite groups

Table 10 provides a pairwise comparison using beta dispersion (Vegan library package), to compare centroid distances from the mean of the Bray-Curtis distances between the malicious versus the benign groups. The observed p-values were calculated from an analytic t-distribution, with the permuted p-values obtained during the 99 permutations against an alpha test of 0.05. This satisfied the parametric assumptions of the t-test by the extremely low p-values for both between-group comparisons.

Next, a Tukey's Honest Significant Differences test was conducted using the permuted analysis of variance functionality, conducted at the 95% family-wise confidence level, resulting in the following output as shown in Table 11, below.

Table 11

Permuted Tukey HDS Test Results

Tukey multiple comparisons of means	*family-wise confidence level*	Fit: aov(formula = distances ~ group, data = df)	$group	diff lwr 85u pr p adj	Malicious-Benign
	95%				-0.08388974
					-0.08871624
					-0.07906324
					0

Note. A statistically significant adjusted p-value of 0.00 against an alpha of 0.05, using the Tukey multiple comparison of means between the Malicious composite group and the Benign composite group classes, based on a permuted analysis of variance model

Table 11 demonstrates the statistically significant (adjusted p-value) of 0 between the centroid means of the Malicious group and the Benign group classes. Based on a permuted analysis of variance (AOV), the test measured the permuted centroid differences between the two groups using mean differences, an upper-end point and a lower-end point, and finally an adjusted p-value. The centroid mean difference between these two groups is statistically significant with an adjusted p-value (permuted p-value) of 0.00 against an alpha of 0.05.

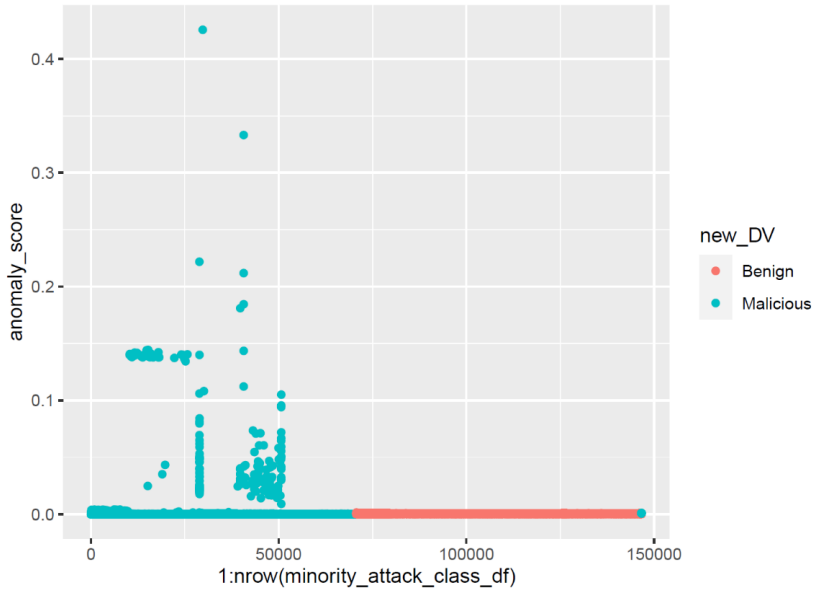
Bland and Altman (1995) explained that the Tukey multiple comparisons of means test is one of several methods that would be applicable to these observations. Also referred to as Tukey's honestly significant difference test, or just simply "Tukey's HSD," this test compares

utilizing an adjustment made for multiple permuted tests conducted within the Vegan library package framework using the R statistical computing environment.

The Tukey HDS test conducts a permuted test similar to an analysis of variance (AOV) between the groups in question, with adjustments made during each permutation. The resulting output provides a “diff” (difference) column for the mean centroid differences, as well as an “lwr” (lower end point of the interval) value and an “upr” value, providing the upper end point of the interval. Finally, the test concludes by providing a permuted adjusted p-value. With our p-value for this test being 0.00 against our stated alpha test statistic of 0.05, our permuted Tukey HSD test is statistically significant at a 95% family-wise confidence level. This provides further evidence in supporting the rejection of the null hypothesis statement for Research Question 4.

Figure 14

Anomaly Scores Scatterplot Matrix



Note. Scatterplot matrix of the anomaly scores produced by the deep learning autoencoder network, showing the anomaly score distributions between the human-labeled composite groups, Benign (Red) and Malicious (Blue)

A color-coded scatterplot matrix generated by plotting the anomaly scores produced by the deep learning autoencoding neural network and color-coded by the binary classification (benign and malicious) based on the human-curated malware class labels in the original data, shown in Figure 14 (above). The vast majority of malicious events achieved an unsupervised anomaly score that was higher than the mean anomaly score for the benign class, without being pre-trained on the human-labeled classification.

Figure 14 provides a color-coded scatterplot of the 150,000 anomaly-scored observations from the statistically balanced binary class of “malicious” and “benign” network flows. The scatterplot provides the percentile anomaly score on the vertical axis as provided by the unsupervised learning autoencoding neural network model, and the observation number of each flow event on the vertical axis.

The research did not incorporate the color-coded class assignment (Benign or Malicious) variable into the self-supervised, untrained autoencoding neural network’s anomaly detector model. Rather, it re-incorporated this class label into the output from the neural network in order to provide a better visualization of how well the neural network detected potentially malicious malware events by providing percentile anomaly scores above the mean for the benign class proportion.

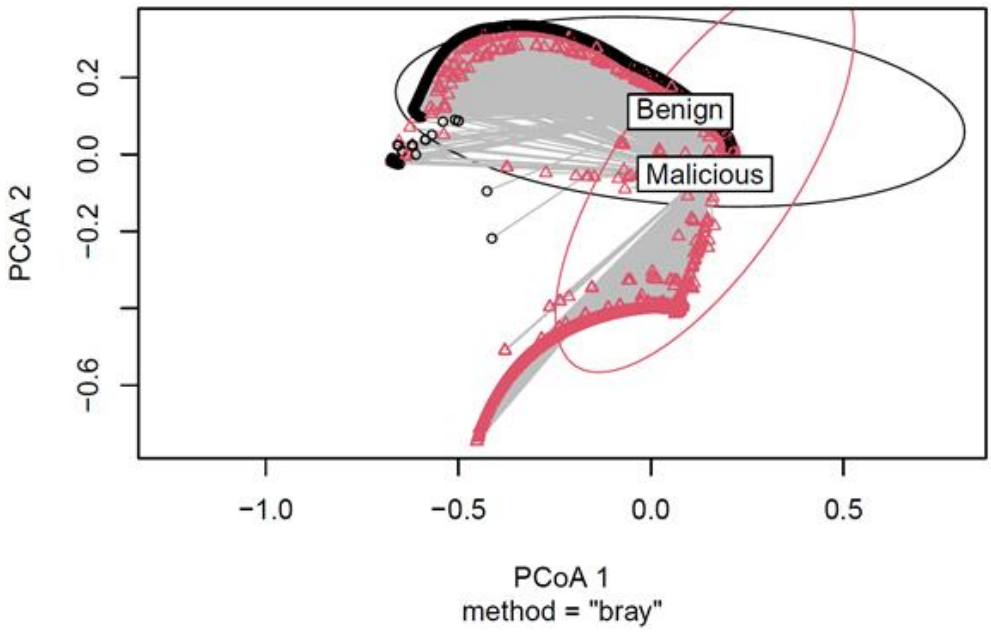
Lastly, Figure 15 (below) provides a plot of the Multivariate Homogeneity of Group Dispersions, produced by plotting the distances to the centroids of the two group labels based on their unsupervised deep learning anomaly score values. This plot occurred using the first two Principal Component of Analysis (PCoA) axes from the transformed output data, using a randomly sampled n=40,000 from the original 150,000 randomly selected flow observations, due to finite available computational resources. The plot utilized ellipses to encapsulate the 90%

confidence interval for the distances to centroids for each group as produced by the beta dispersion methodology test ("betadispr" function) from the Vegan library package in R.

Thus, this plot utilizes permuted non-Euclidean Bray-Curtis distances between samples drawn from n=40,000 randomly selected observations from the original balanced training dataset of 146,684 observations, across the first two principal component axes which incorporated the predicted anomaly scores, demonstrating the relative compactness of each group's distances to centroids for their anomaly scores. This provides further visual support for the permuted statistical significance findings that allowed for the rejection of the null hypothesis for Research Question 4, in favor of the alternative hypothesis.

Figure 15

IoT-23 Multivariate Homogeneity of Group Dispersions



Note. Scatterplot of the Bray-Curtis non-Eucldiean distances from centroids of the autoencoding neural network anomaly score predictions, with the human-labeled binary classes assigned after the anomaly scores have been determined

Figure 15 is a color-coded scatterplot using Bray-Curtis non-Euclidean distances from centroids of the anomaly score predicted values, using the human-labeled binary classes to demonstrate true class memberships. 90% confidence ellipses were plotted over the PCoA dimension scatterplot, showing the relative compactness and distinctiveness of the relative dispersion between the two groups.

Conclusion

In this study, the research specifically addressed findings relevant to the larger field of technology management for digital communication systems cyber defense. This supported the use case for an effective technology management strategy for digital communication systems based on multiple AI/ML defensive cyber strategies through a variety of post hoc experiments. The primary efforts to support technology management were by providing scientific evidence to support the effectiveness of using artificial intelligence and machine learning systems in digital communication networks.

As primary components for an enterprise network defense in-depth strategy to protect against IoT malware attacks, this research proved their effectiveness in mitigating multiple threats. Likewise, it was demonstrated how AI/ML systems can become a proven component in effectively educating technology managers about specific threat classes as they materialize upon the network. In addition, this research proved the significant effectiveness of AI/ML systems in detecting novel malware threats over unrestrained networks for IoT network traffic without having pre-training from human-curated network flows.

The research further demonstrated that it is possible to apply various scientific tests of falsifiability that incorporate reproducible methods into the quasi-experiment design. Most importantly, the research indicated how to apply this falsifiability to the field of machine

learning for the pre-selection of appropriate cyber network flow datasets that could be potential candidates for an AI/ML predictive model. Specifically, the author successfully applied a series of permuted MANOVA techniques to demonstrate statistical significance in the relationship of sequentially added network flow independent variable terms. This scientifically demonstrates against the stated alpha test statistic of 0.05 that a nonlinear relationship exists between the independent variables in the network flows and our dependent set of malware class groups that were contained within an encoded multi-level categorical factor dependent variable representing distinct treatment groups or malware classes, including the benign states. To further support statistical reliability, the number of observations included within the PERMANOVA that tested for statistical significance based on a Chi-square Goodness of Fit Test that incorporated both effect size (w) and statistical power assessments.

The author also made substantial use of tests related to PERMANOVA methods for conducting non-parametric permuted tests of statistical significance, including the use of non-Euclidean Bray-Curtis distance calculations and permuted Tukey HSD scores. The result was a comprehensive display of the ability to conduct AI/ML models for state-of-the-art cyber-defense applications applied to commoditized IoT Raspberry Pi hardware devices and relying solely on (relatively) inexpensive-to-process network flow observations data, rather than the more expensive to store and process packet capture logs. Multivariate Tests of Homogeneity were also extensively utilized to visually and scientifically validate with permuted falsifiability that both pre-trained supervised deep learning predictive models and unsupervised/self-supervised autoencoding deep learning models are both effective instruments for applications to this use case.

The research demonstrated the utility of applying a PERMANOVA model to the raw network flow data prior to being analyzed by any of the AI/ML models, for the purpose of validating that a permuted, statistically significant causal relationship is probabilistically present in the network flow capture data, prior to applying the complex AI/ML predictive modeling frameworks. The author believes that this represents a substantially useful precursor assessment stage for the suitability and reliability of the utilization of any nonparametric statistical learning algorithms applied to cyber network data predictive analytics. This methodology is proposed as a framework to govern costs and development risks (success uncertainty) associated with enterprise-scale AI/ML model development efforts as it applies to cyber network data.

The research further demonstrated that SHapley Additive exPlanations (SHAP values) can be scientifically applied to the detection of malware predictions in network flow data captures for IoT devices, for the purpose of applying HRD principles for training human network subject matter expert staff in complex cyber defense behaviors that are being detected by these AI/ML models.

Recommendations for Future Research

The novelty of utilizing permuted MANOVA methodologies and techniques such as dissimilarities using beta diversity methods to conduct nonparametric transformations of complex multivariate datasets represents an opportunity to demonstrate nonlinear tests of homogeneity in cyber network data. These tests produced a variety of explainable and readily visualized digital signatures for a wide range of cyber network threats.

The author believes that this framework can not only be used to assess the potential applicability for accurate AI/ML supervised malware detection models but may also be used as a

means of falsifiability, determining the statistical significance of these post-hoc experiment models.

The author recommends that further studies involving permuted, nonparametric statistical methods be explored for assessing the potential suitability, accuracy, explainability, and provability of nonparametric statistical learning algorithms to cyber intrusion detection use-case models in both a pre-test and post-test environment to improve utilization efficiency, reliability, and scalability of AI/ML models in this use case.

Likewise, the various permuted tests undertaken, in particular the Bray-Curtis distance calculations used in various tests of homogeneity between groups, are extraordinarily computationally expensive, even on modern computer hardware. Even when using sub-sampled datasets using Goodness of Fit tests to allow sufficient statistical power for the sample size, the permuted tests of homogeneity between groups using these calculated non-Euclidean distance measures often took multiple days (in some instances, nearly a week) on 32-core CPUs running on servers with 64 GB of memory. It would therefore be virtually impossible to run these tests on extremely large network flow dataset universes without the use of sampling.

Optimizing these types of permuted computations, such as migrating them to Spark/Hadoop clusters or GPU computing environments, may be one potential solution for improving processing times by exploiting brute-force hardware acceleration and distributed computing technologies. Likewise, this could allow for the use of larger sample sizes to be potentially practical or even ultimately forgoing the need for sampling entirely. It would also be beneficial when incorporating wider data frames such as deep packet inspection log files.

REFERENCES

Anderson, M. J. (2001). A new method for non-parametric multivariate analysis of variance: Non-parametric Anovac for ecology. *Austral Ecology, 26*(1), 32–46.

Anderson, M. J. (2006). Distance-based tests for homogeneity of multivariate dispersions. *Biometrics, 62*(1), 245–253. https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1541-0420.2005.00440.x

Anderson, M. J. (2017). Permuted multivariate analysis of variance (PERMANOVA). In *Wiley StatsRef: Statistics Reference Online,* (pp. 1–15). American Cancer Society. https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat07841

Anthi, E., Williams, L., Słowińska, M., Theodorakopoulos, G., & Burnap, P. (2019). A supervised intrusion detection system for smart home IoT devices. *IEEE Internet of Things Journal*, *6*(5), 9042–9053.

Banerjee, P., Dehnbostel, F. O., & Preissner, R. (2018). Prediction is a balancing act: Importance of sampling methods to balance sensitivity and specificity of predictive models based on imbalanced chemical data sets. *Frontiers in Chemistry*, 362.

Bland, J. M., & Altman, D. G. (1995). Multiple significance tests: The Bonferroni method. *British Medical Journal*, 310, 170.

Bobrovnikova, K., Lysenko, S., Gaj, P., Martynyuk, V., & Denysiuk, D. (2020). Technique for IoT Cyberattacks Detection Based on DNS Traffic Analysis. In *IntelITSIS* (pp. 208-218).

Broatch, J. E., Dietrich, S., & Goelman, D. (2019). Introducing data science techniques by connecting database concepts and dplyr. *Journal of Statistics Education, 27*(3), 147–153.

Bzdok, D., & Altman, N. (2018). Krzywinski M. *Points of significance: statistics versus machine learning. Nature Methods*, *15*(04), 233-234.

Candel, A., Parmar, V., LeDell, E., & Arora, A. (2016). Deep learning with H2O. *H2O. ai Inc*, 1–21.

Edwards, A. W. (2005). RA Fischer, statistical methods for research workers, (1925). In *Landmark writings in western mathematics 1640-1940* (pp. 856-870). Elsevier Science.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189–1232.

Garcia, S., Parmisano, A., & Erquiaga, M. J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0. 0)[Data set]. Zenodo.

Ganesh, E. N. (2019). Health monitoring system using Raspberry Pi and IoT. *Oriental Journal of Computer Science and Technology, 12*.

Gedeon, T. D. (1997). Data mining of inputs: Analysing magnitude and functional measures. *International Journal of Neural Systems*, *8*(02), 209–218.

Gilley, J., Eggland, S., Gilley, A. M., & Maycunich, A. (2002). *Principles of human resource development*. Basic Books.

Goodfellow, I., Yoshua, B., & Courville, A. (2016). *Deep learning*. MIT press.

Gupta, M. S. D., Patchava, V., & Menezes, V. (2015, October). Healthcare based on IoT using raspberry pi. In *2015 International Conference on Green Computing and Internet of Things* (ICGCIoT; pp. 796–799). IEEE.

Hung, M. (2017). Leading the iot, gartner insights on how to lead in a connected world. *Gartner Research*, *1*, 1-5.

Ibrahim, D. M., Hammoudeh, M. A. A., Ambreen, S., & Mohammadi, S. (2019). Raspberry pi-based smart infant monitoring system. *International Journal of Engineering Research and Technology*, *12*(10), 1723–1729.

ICIRT (2012). Identification of a New Targeted Cyber-Attack. Iran Computer Incident Response Team. Accessed from https://www.webcitation.org/682bfkhaU?url=http://www.certcc.ir/index.php?name=news&file=article&sid=1894&newlang=eng on 12 JUL 2021.

IoT.Business.News. (2022). https://iotbusinessnews.com/2022/05/19/70343-state-of-iot-2022-number-of-connected-iot-devices-growing-18-to-14-4-billion-globally/#:~:text=supply%20chain%20disruptions.-,In%202022%2C%20the%20market%20for%20the%20Internet%20of%20Things%20is,27%20billion%20connected%20IoT%20devices

Jaiswal, K., Sobhanayak, S., Mohanta, B. K., & Jena, D. (2017, November). IoT-cloud based framework for patient's data collection in smart healthcare system using raspberry-pi. In *2017 International conference on electrical and computing technologies and applications (ICECTA)* (pp. 1-4). IEEE.

Jethani, N., Sudarshan, M., Covert, I. C., Lee, S. I., & Ranganath, R. (2021, September). FastSHAP: Real-Time Shapley Value Estimation. In *International Conference on Learning Representations*.

Jolliffe, I. T. (2002). *Principal component analysis for special types of data* (pp. 338-372). Springer New York.

Judson, D. H. (2005). Computerized record linkage and statistical matching. In *Encyclopedia of Social Measurement*. 439-447.

Kaspersky Lab ZAO. (2012). *The flame: Questions and answers*. SECURELIST. Kaspersky Lab Expert. https://www.webcitation.org/68347vKEs?url=https://www.securelist.com/en/blog/208193522/The_Flame_Questions_and_Answers

Keane, P. A., & Topol, E. J. (2018). With an eye to AI and autonomous diagnosis. *NPJ Digital Medicine*, *1*(1), 40.

Kephart, J. O., Sorkin, G. B., Chess, D. M., & White, S. R. (1997). Fighting computer viruses. *Scientific American*, *277*(5), 88-93.

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2018). *Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset*. https://arxiv.org/abs/1811.00701

Lukito, R. B., & Lukito, C. (2019). Development of IoT at hydroponic system using raspberry Pi. *Telkomnika*, *17*(2), 897–906.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

McAlexander, R. J., & Mentch, L. (2020). Predictive inference with random forests: A new perspective on classical analyses. *Research & Politics*, *7*(1), 2053168020905487.

McCarthy, K. (2020). The internet of things is a security nightmare, latest real-world analysis reveals: Unencrypted traffic, network crossover, vulnerable OSs. *The Register*. https://www.theregister.com/2020/03/11/internet_of_things_security_nightmare/

McLean, G. N., & McLean, L. D. (2001). If we can't define HRD in one country, how can we define it in an international context? *Human Resource Development International*, *4*(3), 313–332.

Mehta, A. K., & Patel, P. B. (2019). Smart traffic system using Raspberry-Pi. International Journal of Science & Engineering Development Research (www.ijsdr.org), ISSN:2455-2631, Vol.4, Issue 3, 64-69.

Mohanty, N., John, A. L. S., Manmatha, R., & Rath, T. M. (2013). Shape-based image classification and retrieval. In *Handbook of Statistics* (Vol. 31, pp. 249–267). Elsevier.

Mudaliar, M. D., & Sivakumar, N. (2020). IoT based real time energy monitoring system using Raspberry Pi. *Internet of Things*, *12*, 100292.

Murphy, M. (2017). The Internet of Things and the threat it poses to DNS. *Network Security*, *2017*(7), 17-19.

PaloAlto Networks. (2020). *2020 United 42 IoT threat report*. https://unit42.paloaltonetworks.com/iot-threat-report-2020/

Parmisano, A., Garcia, S., Erquiaga, M. J. (2020). Aposemat IoT-23. A labeled dataset with malicious and benign IoT network traffic. Stratosphere Laboratory. https://www.stratosphereips.org/datasets-iot23

Raj, A., & Steingart, D. (2018). Power sources for the internet of things. *Journal of the Electrochemical Society*, *165*(8), B3130.

Raspberry Pi Foundation GitHub. (2020). https://github.com/raspberrypi/documentation

Raspberry Pi Foundation. (2016). Ten millionth Raspberry Pi, and a new kit. *Raspberry Pi Blog*. https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/

R Core Team. (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing. https://www.R-project.org/

Reshef, D. N., Reshef, Y. A., Finucane, H. K., Grossman, S. R., McVean, G., Turnbaugh, P. J., & Sabeti, P. C. (2011). Detecting novel associations in large data sets. S*cience*, *334*(6062), 1518–1524.

RStudio Team. (2020). RStudio: Integrated development for R. RStudio, PBC. http://www.rstudio.com/

Sengan, S., Khalaf, O. I., Priyadarsini, S., Sharma, D. K., Amarendra, K., & Hamad, A. A. (2022). Smart healthcare security device on medical IoT using raspberry pi. *International Journal of Reliable and Quality E-Healthcare (IJRQEH)*, *11*(3), 1–11.

Shapley, L. S. (1953). A value for n-person games. Contributions to the Theory of Games, *2*(28), 307–317.

Sivaraman, V., Gharakheili, H. H., Fernandes, C., Clark, N., & Karliychuk, T. (2018). Smart IoT devices in the home: Security and privacy implications. *IEEE Technology and Society Magazine*, *37*(2), 71–79.

Stuber, E. F., Chizinski, C. J., Lusk, J. J., & Fontaine, J. J. (2019). Multivariate models and analyses. *Quantitative analyses in wildlife science*, *1*, 32-62.

Szepannek, G. (2022). An overview on the landscape of r packages for open source scorecard modelling. *Risks*, *10*(3), 67.

Tuck, J., & Boyd, S. (2022). Eigen-stratified models. *Optimization and Engineering*, *23*(1), 397-419.

The UNSW-NB15 Dataset. (2020) https://www.unsw.adfa.edu.au/unswcanberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Weber, S. (2004). *The success of open source*. Harvard University Press.

Woolman, T. A., & Lee, S.P. (2020). Network intrusion detection using deep learning and machine learning for multinomial classification. *International Journal of Cyber-Security and Digital Forensics*, *9*(4), 155-182.

Woolman, T. A., & Lunsford, P. (2023). Malware Detection in Network Flows With Self-Supervised Deep Learning. In *Encyclopedia of Data Science and Machine Learning* (pp. 2314-2331). IGI Global.

Xie, Q., Zhang, Q., Zhang, X., Tian, D., Wen, R., Zhu, T., & Li, X. (2021). A context-centric chatbot for cryptocurrency using the bidirectional encoder representations from transformers neural networks. *International Journal of Economics and Management Engineering*, *15*(2), 150–156.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

APPENDIX

R Source Code for all Four Research Question Models

```
##### start here
conn_log_malware_1$timestamp <- NULL
conn_log_malware_1$userid <- NULL
conn_log_malware_1$id.orig_h <- NULL
conn_log_malware_1$id.resp_h <- NULL
conn_log_malware_1$duration <- as.numeric(conn_log_malware_1$duration)
conn_log_malware_1$orig_bytes <- as.numeric(conn_log_malware_1$orig_bytes)
conn_log_malware_1$resp_bytes <- as.numeric(conn_log_malware_1$resp_bytes)
conn_log_malware_1$resp_pkts <- as.numeric(conn_log_malware_1$resp_pkts)
conn_log_malware_1$orig_pkts <- as.numeric(conn_log_malware_1$orig_pkts)
conn_log_malware_1$orig_ip_bytes <- as.numeric(conn_log_malware_1$orig_ip_bytes)
conn_log_malware_1$resp_ip_bytes <- as.numeric(conn_log_malware_1$resp_ip_bytes)

conn_log_malware_1$service <- as.character(conn_log_malware_1$service)
conn_log_malware_1$local_orig <- as.numeric(conn_log_malware_1$local_orig)
conn_log_malware_1$local_resp <- as.numeric(conn_log_malware_1$local_resp)

#for character IVs

conn_log_malware_1[conn_log_malware_1=="-"]<-"0"
#worked :)

conn_log_malware_1$missed_bytes <- as.numeric(conn_log_malware_1$missed_bytes)
conn_log_malware_1[conn_log_malware_1=="(empty)"]<-"0"
conn_log_malware_1$tunnel_parents <- as.numeric(conn_log_malware_1$tunnel_parents)
#conn_log_malware_1[is.na(conn_log_malware_1)] <- 0
conn_log_malware_1$id.orig_p[is.na(conn_log_malware_1$id.orig_p)] <- 0
conn_log_malware_1$id.resp_p[is.na(conn_log_malware_1$id.resp_p)] <- 0
conn_log_malware_1$duration[is.na(conn_log_malware_1$duration)] <- 0
conn_log_malware_1$orig_bytes[is.na(conn_log_malware_1$orig_bytes)] <- 0
conn_log_malware_1$resp_bytes[is.na(conn_log_malware_1$resp_bytes)] <- 0
conn_log_malware_1$local_orig[is.na(conn_log_malware_1$local_orig)] <- 0
conn_log_malware_1$local_resp[is.na(conn_log_malware_1$local_resp)] <- 0
conn_log_malware_1$missed_bytes[is.na(conn_log_malware_1$missed_bytes)] <- 0
conn_log_malware_1$orig_pkts[is.na(conn_log_malware_1$orig_pkts)] <- 0
conn_log_malware_1$orig_ip_bytes[is.na(conn_log_malware_1$orig_ip_bytes)] <- 0
conn_log_malware_1$resp_pkts[is.na(conn_log_malware_1$resp_pkts)] <- 0
conn_log_malware_1$resp_ip_bytes[is.na(conn_log_malware_1$resp_ip_bytes)] <- 0
conn_log_malware_1$tunnel_parents[is.na(conn_log_malware_1$tunnel_parents)] <- 0
conn_log_malware_1$label <- NULL #redundant with the primary DV

#worked!
```

View(conn_log_malware_1)

#####

```
conn_log_malware_3$timestamp <- NULL
conn_log_malware_3$userid <- NULL
conn_log_malware_3$id.orig_h <- NULL
conn_log_malware_3$id.resp_h <- NULL
conn_log_malware_3$duration <- as.numeric(conn_log_malware_3$duration)
conn_log_malware_3$orig_bytes <- as.numeric(conn_log_malware_3$orig_bytes)
conn_log_malware_3$resp_bytes <- as.numeric(conn_log_malware_3$resp_bytes)
conn_log_malware_3$resp_pkts <- as.numeric(conn_log_malware_3$resp_pkts)
conn_log_malware_3$orig_pkts <- as.numeric(conn_log_malware_3$orig_pkts)
conn_log_malware_3$orig_ip_bytes <- as.numeric(conn_log_malware_3$orig_ip_bytes)
conn_log_malware_3$resp_ip_bytes <- as.numeric(conn_log_malware_3$resp_ip_bytes)

conn_log_malware_3$service <- as.character(conn_log_malware_3$service)
conn_log_malware_3$local_orig <- as.numeric(conn_log_malware_3$local_orig)
conn_log_malware_3$local_resp <- as.numeric(conn_log_malware_3$local_resp)

#for character IVs

conn_log_malware_3[conn_log_malware_3=="-"]<-"0"


conn_log_malware_3$missed_bytes <- as.numeric(conn_log_malware_3$missed_bytes)
conn_log_malware_3[conn_log_malware_3=="(empty)"]<-"0"
conn_log_malware_3$tunnel_parents <- as.numeric(conn_log_malware_3$tunnel_parents)
#conn_log_malware_3[is.na(conn_log_malware_3)] <- 0
conn_log_malware_3$id.orig_p[is.na(conn_log_malware_3$id.orig_p)] <- 0
conn_log_malware_3$id.resp_p[is.na(conn_log_malware_3$id.resp_p)] <- 0
conn_log_malware_3$duration[is.na(conn_log_malware_3$duration)] <- 0
conn_log_malware_3$orig_bytes[is.na(conn_log_malware_3$orig_bytes)] <- 0
conn_log_malware_3$resp_bytes[is.na(conn_log_malware_3$resp_bytes)] <- 0
conn_log_malware_3$local_orig[is.na(conn_log_malware_3$local_orig)] <- 0
conn_log_malware_3$local_resp[is.na(conn_log_malware_3$local_resp)] <- 0
conn_log_malware_3$missed_bytes[is.na(conn_log_malware_3$missed_bytes)] <- 0
conn_log_malware_3$orig_pkts[is.na(conn_log_malware_3$orig_pkts)] <- 0
conn_log_malware_3$orig_ip_bytes[is.na(conn_log_malware_3$orig_ip_bytes)] <- 0
conn_log_malware_3$resp_pkts[is.na(conn_log_malware_3$resp_pkts)] <- 0
conn_log_malware_3$resp_ip_bytes[is.na(conn_log_malware_3$resp_ip_bytes)] <- 0
conn_log_malware_3$tunnel_parents[is.na(conn_log_malware_3$tunnel_parents)] <- 0
conn_log_malware_3$label <- NULL #redundant with the primary DV

#worked!
```

```
View(conn_log_malware_3)

###

conn_log_malware_7$timestamp <- NULL
conn_log_malware_7$userid <- NULL
conn_log_malware_7$id.orig_h <- NULL
conn_log_malware_7$id.resp_h <- NULL
conn_log_malware_7$duration <- as.numeric(conn_log_malware_7$duration)
conn_log_malware_7$orig_bytes <- as.numeric(conn_log_malware_7$orig_bytes)
conn_log_malware_7$resp_bytes <- as.numeric(conn_log_malware_7$resp_bytes)
conn_log_malware_7$resp_pkts <- as.numeric(conn_log_malware_7$resp_pkts)
conn_log_malware_7$orig_pkts <- as.numeric(conn_log_malware_7$orig_pkts)
conn_log_malware_7$orig_ip_bytes <- as.numeric(conn_log_malware_7$orig_ip_bytes)
conn_log_malware_7$resp_ip_bytes <- as.numeric(conn_log_malware_7$resp_ip_bytes)

conn_log_malware_7$service <- as.character(conn_log_malware_7$service)
conn_log_malware_7$local_orig <- as.numeric(conn_log_malware_7$local_orig)
conn_log_malware_7$local_resp <- as.numeric(conn_log_malware_7$local_resp)

#for character IVs

conn_log_malware_3[conn_log_malware_3=="-"]<-"0"


conn_log_malware_7$missed_bytes <- as.numeric(conn_log_malware_7$missed_bytes)
conn_log_malware_7[conn_log_malware_7=="(empty)"]<-"0"
conn_log_malware_7$tunnel_parents <- as.numeric(conn_log_malware_7$tunnel_parents)
#conn_log_malware_7[is.na(conn_log_malware_7)] <- 0
conn_log_malware_7$id.orig_p[is.na(conn_log_malware_7$id.orig_p)] <- 0
conn_log_malware_7$id.resp_p[is.na(conn_log_malware_7$id.resp_p)] <- 0
conn_log_malware_7$duration[is.na(conn_log_malware_7$duration)] <- 0
conn_log_malware_7$orig_bytes[is.na(conn_log_malware_7$orig_bytes)] <- 0
conn_log_malware_7$resp_bytes[is.na(conn_log_malware_7$resp_bytes)] <- 0
conn_log_malware_7$local_orig[is.na(conn_log_malware_7$local_orig)] <- 0
conn_log_malware_7$local_resp[is.na(conn_log_malware_7$local_resp)] <- 0
conn_log_malware_7$missed_bytes[is.na(conn_log_malware_7$missed_bytes)] <- 0
conn_log_malware_7$orig_pkts[is.na(conn_log_malware_7$orig_pkts)] <- 0
conn_log_malware_7$orig_ip_bytes[is.na(conn_log_malware_7$orig_ip_bytes)] <- 0
conn_log_malware_7$resp_pkts[is.na(conn_log_malware_7$resp_pkts)] <- 0
conn_log_malware_7$resp_ip_bytes[is.na(conn_log_malware_7$resp_ip_bytes)] <- 0
conn_log_malware_7$tunnel_parents[is.na(conn_log_malware_7$tunnel_parents)] <- 0
conn_log_malware_7$label <- NULL #redundant with the primary DV

#worked!
```

View(conn_log_malware_7)

###

```
conn_log_malware_8$timestamp <- NULL
conn_log_malware_8$userid <- NULL
conn_log_malware_8$id.orig_h <- NULL
conn_log_malware_8$id.resp_h <- NULL
conn_log_malware_8$duration <- as.numeric(conn_log_malware_8$duration)
conn_log_malware_8$orig_bytes <- as.numeric(conn_log_malware_8$orig_bytes)
conn_log_malware_8$resp_bytes <- as.numeric(conn_log_malware_8$resp_bytes)
conn_log_malware_8$resp_pkts <- as.numeric(conn_log_malware_8$resp_pkts)
conn_log_malware_8$orig_pkts <- as.numeric(conn_log_malware_8$orig_pkts)
conn_log_malware_8$orig_ip_bytes <- as.numeric(conn_log_malware_8$orig_ip_bytes)
conn_log_malware_8$resp_ip_bytes <- as.numeric(conn_log_malware_8$resp_ip_bytes)

conn_log_malware_8$service <- as.character(conn_log_malware_8$service)
conn_log_malware_8$local_orig <- as.numeric(conn_log_malware_8$local_orig)
conn_log_malware_8$local_resp <- as.numeric(conn_log_malware_8$local_resp)

#for character IVs

conn_log_malware_8[conn_log_malware_8=="-"]<-"0"


conn_log_malware_8$missed_bytes <- as.numeric(conn_log_malware_8$missed_bytes)
conn_log_malware_8[conn_log_malware_8=="(empty)"]<-"0"
conn_log_malware_8$tunnel_parents <- as.numeric(conn_log_malware_8$tunnel_parents)
#conn_log_malware_8[is.na(conn_log_malware_8)] <- 0
conn_log_malware_8$id.orig_p[is.na(conn_log_malware_8$id.orig_p)] <- 0
conn_log_malware_8$id.resp_p[is.na(conn_log_malware_8$id.resp_p)] <- 0
conn_log_malware_8$duration[is.na(conn_log_malware_8$duration)] <- 0
conn_log_malware_8$orig_bytes[is.na(conn_log_malware_8$orig_bytes)] <- 0
conn_log_malware_8$resp_bytes[is.na(conn_log_malware_8$resp_bytes)] <- 0
conn_log_malware_8$local_orig[is.na(conn_log_malware_8$local_orig)] <- 0
conn_log_malware_8$local_resp[is.na(conn_log_malware_8$local_resp)] <- 0
conn_log_malware_8$missed_bytes[is.na(conn_log_malware_8$missed_bytes)] <- 0
conn_log_malware_8$orig_pkts[is.na(conn_log_malware_8$orig_pkts)] <- 0
conn_log_malware_8$orig_ip_bytes[is.na(conn_log_malware_8$orig_ip_bytes)] <- 0
conn_log_malware_8$resp_pkts[is.na(conn_log_malware_8$resp_pkts)] <- 0
conn_log_malware_8$resp_ip_bytes[is.na(conn_log_malware_8$resp_ip_bytes)] <- 0
conn_log_malware_8$tunnel_parents[is.na(conn_log_malware_8$tunnel_parents)] <- 0
conn_log_malware_8$label <- NULL #redundant with the primary DV
```

```
#worked!

View(conn_log_malware_8)



###

###

conn_log_malware_9$timestamp <- NULL
conn_log_malware_9$userid <- NULL
conn_log_malware_9$id.orig_h <- NULL
conn_log_malware_9$id.resp_h <- NULL
conn_log_malware_9$duration <- as.numeric(conn_log_malware_9$duration)
conn_log_malware_9$orig_bytes <- as.numeric(conn_log_malware_9$orig_bytes)
conn_log_malware_9$resp_bytes <- as.numeric(conn_log_malware_9$resp_bytes)
conn_log_malware_9$resp_pkts <- as.numeric(conn_log_malware_9$resp_pkts)
conn_log_malware_9$orig_pkts <- as.numeric(conn_log_malware_9$orig_pkts)
conn_log_malware_9$orig_ip_bytes <- as.numeric(conn_log_malware_9$orig_ip_bytes)
conn_log_malware_9$resp_ip_bytes <- as.numeric(conn_log_malware_9$resp_ip_bytes)

conn_log_malware_9$service <- as.character(conn_log_malware_9$service)
conn_log_malware_9$local_orig <- as.numeric(conn_log_malware_9$local_orig)
conn_log_malware_9$local_resp <- as.numeric(conn_log_malware_9$local_resp)

#for character IVs

conn_log_malware_9[conn_log_malware_9=="-"]<-"0"


conn_log_malware_9$missed_bytes <- as.numeric(conn_log_malware_9$missed_bytes)
conn_log_malware_9[conn_log_malware_9=="(empty)"]<-"0"
conn_log_malware_9$tunnel_parents <- as.numeric(conn_log_malware_9$tunnel_parents)
#conn_log_malware_9[is.na(conn_log_malware_9)] <- 0
conn_log_malware_9$id.orig_p[is.na(conn_log_malware_9$id.orig_p)] <- 0
conn_log_malware_9$id.resp_p[is.na(conn_log_malware_9$id.resp_p)] <- 0
conn_log_malware_9$duration[is.na(conn_log_malware_9$duration)] <- 0
conn_log_malware_9$orig_bytes[is.na(conn_log_malware_9$orig_bytes)] <- 0
conn_log_malware_9$resp_bytes[is.na(conn_log_malware_9$resp_bytes)] <- 0
conn_log_malware_9$local_orig[is.na(conn_log_malware_9$local_orig)] <- 0
conn_log_malware_9$local_resp[is.na(conn_log_malware_9$local_resp)] <- 0
conn_log_malware_9$missed_bytes[is.na(conn_log_malware_9$missed_bytes)] <- 0
conn_log_malware_9$orig_pkts[is.na(conn_log_malware_9$orig_pkts)] <- 0
conn_log_malware_9$orig_ip_bytes[is.na(conn_log_malware_9$orig_ip_bytes)] <- 0
```

```
conn_log_malware_9$resp_pkts[is.na(conn_log_malware_9$resp_pkts)] <- 0
conn_log_malware_9$resp_ip_bytes[is.na(conn_log_malware_9$resp_ip_bytes)] <- 0
conn_log_malware_9$tunnel_parents[is.na(conn_log_malware_9$tunnel_parents)] <- 0
conn_log_malware_9$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_9)



###


###

conn_log_malware_17$timestamp <- NULL
conn_log_malware_17$userid <- NULL
conn_log_malware_17$id.orig_h <- NULL
conn_log_malware_17$id.resp_h <- NULL
conn_log_malware_17$duration <- as.numeric(conn_log_malware_17$duration)
conn_log_malware_17$orig_bytes <- as.numeric(conn_log_malware_17$orig_bytes)
conn_log_malware_17$resp_bytes <- as.numeric(conn_log_malware_17$resp_bytes)
conn_log_malware_17$resp_pkts <- as.numeric(conn_log_malware_17$resp_pkts)
conn_log_malware_17$orig_pkts <- as.numeric(conn_log_malware_17$orig_pkts)
conn_log_malware_17$orig_ip_bytes <- as.numeric(conn_log_malware_17$orig_ip_bytes)
conn_log_malware_17$resp_ip_bytes <- as.numeric(conn_log_malware_17$resp_ip_bytes)

conn_log_malware_17$service <- as.character(conn_log_malware_17$service)
conn_log_malware_17$local_orig <- as.numeric(conn_log_malware_17$local_orig)
conn_log_malware_17$local_resp <- as.numeric(conn_log_malware_17$local_resp)

#for character IVs

conn_log_malware_17[conn_log_malware_17=="-"]<-"0"


conn_log_malware_17$missed_bytes <- as.numeric(conn_log_malware_17$missed_bytes)
conn_log_malware_17[conn_log_malware_17=="(empty)"]<-"0"
conn_log_malware_17$tunnel_parents <- as.numeric(conn_log_malware_17$tunnel_parents)
#conn_log_malware_17[is.na(conn_log_malware_17)] <- 0
conn_log_malware_17$id.orig_p[is.na(conn_log_malware_17$id.orig_p)] <- 0
conn_log_malware_17$id.resp_p[is.na(conn_log_malware_17$id.resp_p)] <- 0
conn_log_malware_17$duration[is.na(conn_log_malware_17$duration)] <- 0
conn_log_malware_17$orig_bytes[is.na(conn_log_malware_17$orig_bytes)] <- 0
conn_log_malware_17$resp_bytes[is.na(conn_log_malware_17$resp_bytes)] <- 0
```

```
conn_log_malware_17$local_orig[is.na(conn_log_malware_17$local_orig)] <- 0
conn_log_malware_17$local_resp[is.na(conn_log_malware_17$local_resp)] <- 0
conn_log_malware_17$missed_bytes[is.na(conn_log_malware_17$missed_bytes)] <- 0
conn_log_malware_17$orig_pkts[is.na(conn_log_malware_17$orig_pkts)] <- 0
conn_log_malware_17$orig_ip_bytes[is.na(conn_log_malware_17$orig_ip_bytes)] <- 0
conn_log_malware_17$resp_pkts[is.na(conn_log_malware_17$resp_pkts)] <- 0
conn_log_malware_17$resp_ip_bytes[is.na(conn_log_malware_17$resp_ip_bytes)] <- 0
conn_log_malware_17$tunnel_parents[is.na(conn_log_malware_17$tunnel_parents)] <- 0
conn_log_malware_17$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_17)


###

###

conn_log_malware_20$timestamp <- NULL
conn_log_malware_20$userid <- NULL
conn_log_malware_20$id.orig_h <- NULL
conn_log_malware_20$id.resp_h <- NULL
conn_log_malware_20$duration <- as.numeric(conn_log_malware_20$duration)
conn_log_malware_20$orig_bytes <- as.numeric(conn_log_malware_20$orig_bytes)
conn_log_malware_20$resp_bytes <- as.numeric(conn_log_malware_20$resp_bytes)
conn_log_malware_20$resp_pkts <- as.numeric(conn_log_malware_20$resp_pkts)
conn_log_malware_20$orig_pkts <- as.numeric(conn_log_malware_20$orig_pkts)
conn_log_malware_20$orig_ip_bytes <- as.numeric(conn_log_malware_20$orig_ip_bytes)
conn_log_malware_20$resp_ip_bytes <- as.numeric(conn_log_malware_20$resp_ip_bytes)

conn_log_malware_20$service <- as.character(conn_log_malware_20$service)
conn_log_malware_20$local_orig <- as.numeric(conn_log_malware_20$local_orig)
conn_log_malware_20$local_resp <- as.numeric(conn_log_malware_20$local_resp)

#for character IVs

conn_log_malware_20[conn_log_malware_20=="-"]<-"0"


conn_log_malware_20$missed_bytes <- as.numeric(conn_log_malware_20$missed_bytes)
conn_log_malware_20[conn_log_malware_20=="(empty)"]<-"0"
conn_log_malware_20$tunnel_parents <- as.numeric(conn_log_malware_20$tunnel_parents)
#conn_log_malware_20[is.na(conn_log_malware_20)] <- 0
conn_log_malware_20$id.orig_p[is.na(conn_log_malware_20$id.orig_p)] <- 0
conn_log_malware_20$id.resp_p[is.na(conn_log_malware_20$id.resp_p)] <- 0
```

```
conn_log_malware_20$duration[is.na(conn_log_malware_20$duration)] <- 0
conn_log_malware_20$orig_bytes[is.na(conn_log_malware_20$orig_bytes)] <- 0
conn_log_malware_20$resp_bytes[is.na(conn_log_malware_20$resp_bytes)] <- 0
conn_log_malware_20$local_orig[is.na(conn_log_malware_20$local_orig)] <- 0
conn_log_malware_20$local_resp[is.na(conn_log_malware_20$local_resp)] <- 0
conn_log_malware_20$missed_bytes[is.na(conn_log_malware_20$missed_bytes)] <- 0
conn_log_malware_20$orig_pkts[is.na(conn_log_malware_20$orig_pkts)] <- 0
conn_log_malware_20$orig_ip_bytes[is.na(conn_log_malware_20$orig_ip_bytes)] <- 0
conn_log_malware_20$resp_pkts[is.na(conn_log_malware_20$resp_pkts)] <- 0
conn_log_malware_20$resp_ip_bytes[is.na(conn_log_malware_20$resp_ip_bytes)] <- 0
conn_log_malware_20$tunnel_parents[is.na(conn_log_malware_20$tunnel_parents)] <- 0
conn_log_malware_20$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_20)

###


conn_log_malware_21$timestamp <- NULL
conn_log_malware_21$userid <- NULL
conn_log_malware_21$id.orig_h <- NULL
conn_log_malware_21$id.resp_h <- NULL
conn_log_malware_21$duration <- as.numeric(conn_log_malware_21$duration)
conn_log_malware_21$orig_bytes <- as.numeric(conn_log_malware_21$orig_bytes)
conn_log_malware_21$resp_bytes <- as.numeric(conn_log_malware_21$resp_bytes)
conn_log_malware_21$resp_pkts <- as.numeric(conn_log_malware_21$resp_pkts)
conn_log_malware_21$orig_pkts <- as.numeric(conn_log_malware_21$orig_pkts)
conn_log_malware_21$orig_ip_bytes <- as.numeric(conn_log_malware_21$orig_ip_bytes)
conn_log_malware_21$resp_ip_bytes <- as.numeric(conn_log_malware_21$resp_ip_bytes)

conn_log_malware_21$service <- as.character(conn_log_malware_21$service)
conn_log_malware_21$local_orig <- as.numeric(conn_log_malware_21$local_orig)
conn_log_malware_21$local_resp <- as.numeric(conn_log_malware_21$local_resp)

#for character IVs

conn_log_malware_21[conn_log_malware_21=="-"]<-"0"


conn_log_malware_21$missed_bytes <- as.numeric(conn_log_malware_21$missed_bytes)
conn_log_malware_21[conn_log_malware_21=="(empty)"]<-"0"
conn_log_malware_21$tunnel_parents <- as.numeric(conn_log_malware_21$tunnel_parents)
#conn_log_malware_21[is.na(conn_log_malware_21)] <- 0
conn_log_malware_21$id.orig_p[is.na(conn_log_malware_21$id.orig_p)] <- 0
```

```
conn_log_malware_21$id.resp_p[is.na(conn_log_malware_21$id.resp_p)] <- 0
conn_log_malware_21$duration[is.na(conn_log_malware_21$duration)] <- 0
conn_log_malware_21$orig_bytes[is.na(conn_log_malware_21$orig_bytes)] <- 0
conn_log_malware_21$resp_bytes[is.na(conn_log_malware_21$resp_bytes)] <- 0
conn_log_malware_21$local_orig[is.na(conn_log_malware_21$local_orig)] <- 0
conn_log_malware_21$local_resp[is.na(conn_log_malware_21$local_resp)] <- 0
conn_log_malware_21$missed_bytes[is.na(conn_log_malware_21$missed_bytes)] <- 0
conn_log_malware_21$orig_pkts[is.na(conn_log_malware_21$orig_pkts)] <- 0
conn_log_malware_21$orig_ip_bytes[is.na(conn_log_malware_21$orig_ip_bytes)] <- 0
conn_log_malware_21$resp_pkts[is.na(conn_log_malware_21$resp_pkts)] <- 0
conn_log_malware_21$resp_ip_bytes[is.na(conn_log_malware_21$resp_ip_bytes)] <- 0
conn_log_malware_21$tunnel_parents[is.na(conn_log_malware_21$tunnel_parents)] <- 0
conn_log_malware_21$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_21)

###


###


conn_log_malware_33$timestamp <- NULL
conn_log_malware_33$userid <- NULL
conn_log_malware_33$id.orig_h <- NULL
conn_log_malware_33$id.resp_h <- NULL
conn_log_malware_33$duration <- as.numeric(conn_log_malware_33$duration)
conn_log_malware_33$orig_bytes <- as.numeric(conn_log_malware_33$orig_bytes)
conn_log_malware_33$resp_bytes <- as.numeric(conn_log_malware_33$resp_bytes)
conn_log_malware_33$resp_pkts <- as.numeric(conn_log_malware_33$resp_pkts)
conn_log_malware_33$orig_pkts <- as.numeric(conn_log_malware_33$orig_pkts)
conn_log_malware_33$orig_ip_bytes <- as.numeric(conn_log_malware_33$orig_ip_bytes)
conn_log_malware_33$resp_ip_bytes <- as.numeric(conn_log_malware_33$resp_ip_bytes)

conn_log_malware_33$service <- as.character(conn_log_malware_33$service)
conn_log_malware_33$local_orig <- as.numeric(conn_log_malware_33$local_orig)
conn_log_malware_33$local_resp <- as.numeric(conn_log_malware_33$local_resp)

#for character IVs

conn_log_malware_33[conn_log_malware_33=="-"]<-"0"


conn_log_malware_33$missed_bytes <- as.numeric(conn_log_malware_33$missed_bytes)
```

```
conn_log_malware_33[conn_log_malware_33=="(empty)"]<-"0"
conn_log_malware_33$tunnel_parents <- as.numeric(conn_log_malware_33$tunnel_parents)
#conn_log_malware_33[is.na(conn_log_malware_33)] <- 0
conn_log_malware_33$id.orig_p[is.na(conn_log_malware_33$id.orig_p)] <- 0
conn_log_malware_33$id.resp_p[is.na(conn_log_malware_33$id.resp_p)] <- 0
conn_log_malware_33$duration[is.na(conn_log_malware_33$duration)] <- 0
conn_log_malware_33$orig_bytes[is.na(conn_log_malware_33$orig_bytes)] <- 0
conn_log_malware_33$resp_bytes[is.na(conn_log_malware_33$resp_bytes)] <- 0
conn_log_malware_33$local_orig[is.na(conn_log_malware_33$local_orig)] <- 0
conn_log_malware_33$local_resp[is.na(conn_log_malware_33$local_resp)] <- 0
conn_log_malware_33$missed_bytes[is.na(conn_log_malware_33$missed_bytes)] <- 0
conn_log_malware_33$orig_pkts[is.na(conn_log_malware_33$orig_pkts)] <- 0
conn_log_malware_33$orig_ip_bytes[is.na(conn_log_malware_33$orig_ip_bytes)] <- 0
conn_log_malware_33$resp_pkts[is.na(conn_log_malware_33$resp_pkts)] <- 0
conn_log_malware_33$resp_ip_bytes[is.na(conn_log_malware_33$resp_ip_bytes)] <- 0
conn_log_malware_33$tunnel_parents[is.na(conn_log_malware_33$tunnel_parents)] <- 0
conn_log_malware_33$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_33)

###


###


conn_log_malware_34$timestamp <- NULL
conn_log_malware_34$userid <- NULL
conn_log_malware_34$id.orig_h <- NULL
conn_log_malware_34$id.resp_h <- NULL
conn_log_malware_34$duration <- as.numeric(conn_log_malware_34$duration)
conn_log_malware_34$orig_bytes <- as.numeric(conn_log_malware_34$orig_bytes)
conn_log_malware_34$resp_bytes <- as.numeric(conn_log_malware_34$resp_bytes)
conn_log_malware_34$resp_pkts <- as.numeric(conn_log_malware_34$resp_pkts)
conn_log_malware_34$orig_pkts <- as.numeric(conn_log_malware_34$orig_pkts)
conn_log_malware_34$orig_ip_bytes <- as.numeric(conn_log_malware_34$orig_ip_bytes)
conn_log_malware_34$resp_ip_bytes <- as.numeric(conn_log_malware_34$resp_ip_bytes)

conn_log_malware_34$service <- as.character(conn_log_malware_34$service)
conn_log_malware_34$local_orig <- as.numeric(conn_log_malware_34$local_orig)
conn_log_malware_34$local_resp <- as.numeric(conn_log_malware_34$local_resp)

#for character IVs
```

```
conn_log_malware_34[conn_log_malware_34=="-"]<-"0"


conn_log_malware_34$missed_bytes <- as.numeric(conn_log_malware_34$missed_bytes)
conn_log_malware_34[conn_log_malware_34=="(empty)"]<-"0"
conn_log_malware_34$tunnel_parents <- as.numeric(conn_log_malware_34$tunnel_parents)
#conn_log_malware_34[is.na(conn_log_malware_34)] <- 0
conn_log_malware_34$id.orig_p[is.na(conn_log_malware_34$id.orig_p)] <- 0
conn_log_malware_34$id.resp_p[is.na(conn_log_malware_34$id.resp_p)] <- 0
conn_log_malware_34$duration[is.na(conn_log_malware_34$duration)] <- 0
conn_log_malware_34$orig_bytes[is.na(conn_log_malware_34$orig_bytes)] <- 0
conn_log_malware_34$resp_bytes[is.na(conn_log_malware_34$resp_bytes)] <- 0
conn_log_malware_34$local_orig[is.na(conn_log_malware_34$local_orig)] <- 0
conn_log_malware_34$local_resp[is.na(conn_log_malware_34$local_resp)] <- 0
conn_log_malware_34$missed_bytes[is.na(conn_log_malware_34$missed_bytes)] <- 0
conn_log_malware_34$orig_pkts[is.na(conn_log_malware_34$orig_pkts)] <- 0
conn_log_malware_34$orig_ip_bytes[is.na(conn_log_malware_34$orig_ip_bytes)] <- 0
conn_log_malware_34$resp_pkts[is.na(conn_log_malware_34$resp_pkts)] <- 0
conn_log_malware_34$resp_ip_bytes[is.na(conn_log_malware_34$resp_ip_bytes)] <- 0
conn_log_malware_34$tunnel_parents[is.na(conn_log_malware_34$tunnel_parents)] <- 0
conn_log_malware_34$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_34)

###


###


conn_log_malware_35$timestamp <- NULL
conn_log_malware_35$userid <- NULL
conn_log_malware_35$id.orig_h <- NULL
conn_log_malware_35$id.resp_h <- NULL
conn_log_malware_35$duration <- as.numeric(conn_log_malware_35$duration)
conn_log_malware_35$orig_bytes <- as.numeric(conn_log_malware_35$orig_bytes)
conn_log_malware_35$resp_bytes <- as.numeric(conn_log_malware_35$resp_bytes)
conn_log_malware_35$resp_pkts <- as.numeric(conn_log_malware_35$resp_pkts)
conn_log_malware_35$orig_pkts <- as.numeric(conn_log_malware_35$orig_pkts)
conn_log_malware_35$orig_ip_bytes <- as.numeric(conn_log_malware_35$orig_ip_bytes)
conn_log_malware_35$resp_ip_bytes <- as.numeric(conn_log_malware_35$resp_ip_bytes)

conn_log_malware_35$service <- as.character(conn_log_malware_35$service)
conn_log_malware_35$local_orig <- as.numeric(conn_log_malware_35$local_orig)
```

```
conn_log_malware_35$local_resp <- as.numeric(conn_log_malware_35$local_resp)

#for character IVs

conn_log_malware_35[conn_log_malware_35=="-"]<-"0"


conn_log_malware_35$missed_bytes <- as.numeric(conn_log_malware_35$missed_bytes)
conn_log_malware_35[conn_log_malware_35=="(empty)"]<-"0"
conn_log_malware_35$tunnel_parents <- as.numeric(conn_log_malware_35$tunnel_parents)
#conn_log_malware_35[is.na(conn_log_malware_35)] <- 0
conn_log_malware_35$id.orig_p[is.na(conn_log_malware_35$id.orig_p)] <- 0
conn_log_malware_35$id.resp_p[is.na(conn_log_malware_35$id.resp_p)] <- 0
conn_log_malware_35$duration[is.na(conn_log_malware_35$duration)] <- 0
conn_log_malware_35$orig_bytes[is.na(conn_log_malware_35$orig_bytes)] <- 0
conn_log_malware_35$resp_bytes[is.na(conn_log_malware_35$resp_bytes)] <- 0
conn_log_malware_35$local_orig[is.na(conn_log_malware_35$local_orig)] <- 0
conn_log_malware_35$local_resp[is.na(conn_log_malware_35$local_resp)] <- 0
conn_log_malware_35$missed_bytes[is.na(conn_log_malware_35$missed_bytes)] <- 0
conn_log_malware_35$orig_pkts[is.na(conn_log_malware_35$orig_pkts)] <- 0
conn_log_malware_35$orig_ip_bytes[is.na(conn_log_malware_35$orig_ip_bytes)] <- 0
conn_log_malware_35$resp_pkts[is.na(conn_log_malware_35$resp_pkts)] <- 0
conn_log_malware_35$resp_ip_bytes[is.na(conn_log_malware_35$resp_ip_bytes)] <- 0
conn_log_malware_35$tunnel_parents[is.na(conn_log_malware_35$tunnel_parents)] <- 0
conn_log_malware_35$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_35)

###


###


conn_log_malware_36$timestamp <- NULL
conn_log_malware_36$userid <- NULL
conn_log_malware_36$id.orig_h <- NULL
conn_log_malware_36$id.resp_h <- NULL
conn_log_malware_36$duration <- as.numeric(conn_log_malware_36$duration)
conn_log_malware_36$orig_bytes <- as.numeric(conn_log_malware_36$orig_bytes)
conn_log_malware_36$resp_bytes <- as.numeric(conn_log_malware_36$resp_bytes)
conn_log_malware_36$resp_pkts <- as.numeric(conn_log_malware_36$resp_pkts)
conn_log_malware_36$orig_pkts <- as.numeric(conn_log_malware_36$orig_pkts)
conn_log_malware_36$orig_ip_bytes <- as.numeric(conn_log_malware_36$orig_ip_bytes)
```

```
conn_log_malware_36$resp_ip_bytes <- as.numeric(conn_log_malware_36$resp_ip_bytes)

conn_log_malware_36$service <- as.character(conn_log_malware_36$service)
conn_log_malware_36$local_orig <- as.numeric(conn_log_malware_36$local_orig)
conn_log_malware_36$local_resp <- as.numeric(conn_log_malware_36$local_resp)

#for character IVs

conn_log_malware_36[conn_log_malware_36=="-"]<-"0"


conn_log_malware_36$missed_bytes <- as.numeric(conn_log_malware_36$missed_bytes)
conn_log_malware_36[conn_log_malware_36=="(empty)"]<-"0"
conn_log_malware_36$tunnel_parents <- as.numeric(conn_log_malware_36$tunnel_parents)
#conn_log_malware_36[is.na(conn_log_malware_36)] <- 0
conn_log_malware_36$id.orig_p[is.na(conn_log_malware_36$id.orig_p)] <- 0
conn_log_malware_36$id.resp_p[is.na(conn_log_malware_36$id.resp_p)] <- 0
conn_log_malware_36$duration[is.na(conn_log_malware_36$duration)] <- 0
conn_log_malware_36$orig_bytes[is.na(conn_log_malware_36$orig_bytes)] <- 0
conn_log_malware_36$resp_bytes[is.na(conn_log_malware_36$resp_bytes)] <- 0
conn_log_malware_36$local_orig[is.na(conn_log_malware_36$local_orig)] <- 0
conn_log_malware_36$local_resp[is.na(conn_log_malware_36$local_resp)] <- 0
conn_log_malware_36$missed_bytes[is.na(conn_log_malware_36$missed_bytes)] <- 0
conn_log_malware_36$orig_pkts[is.na(conn_log_malware_36$orig_pkts)] <- 0
conn_log_malware_36$orig_ip_bytes[is.na(conn_log_malware_36$orig_ip_bytes)] <- 0
conn_log_malware_36$resp_pkts[is.na(conn_log_malware_36$resp_pkts)] <- 0
conn_log_malware_36$resp_ip_bytes[is.na(conn_log_malware_36$resp_ip_bytes)] <- 0
conn_log_malware_36$tunnel_parents[is.na(conn_log_malware_36$tunnel_parents)] <- 0
conn_log_malware_36$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_36)

###


###


conn_log_malware_39$timestamp <- NULL
conn_log_malware_39$userid <- NULL
conn_log_malware_39$id.orig_h <- NULL
conn_log_malware_39$id.resp_h <- NULL
conn_log_malware_39$duration <- as.numeric(conn_log_malware_39$duration)
conn_log_malware_39$orig_bytes <- as.numeric(conn_log_malware_39$orig_bytes)
```

```
conn_log_malware_39$resp_bytes <- as.numeric(conn_log_malware_39$resp_bytes)
conn_log_malware_39$resp_pkts <- as.numeric(conn_log_malware_39$resp_pkts)
conn_log_malware_39$orig_pkts <- as.numeric(conn_log_malware_39$orig_pkts)
conn_log_malware_39$orig_ip_bytes <- as.numeric(conn_log_malware_39$orig_ip_bytes)
conn_log_malware_39$resp_ip_bytes <- as.numeric(conn_log_malware_39$resp_ip_bytes)

conn_log_malware_39$service <- as.character(conn_log_malware_39$service)
conn_log_malware_39$local_orig <- as.numeric(conn_log_malware_39$local_orig)
conn_log_malware_39$local_resp <- as.numeric(conn_log_malware_39$local_resp)

#for character IVs

conn_log_malware_39[conn_log_malware_39=="-"]<-"0"


conn_log_malware_39$missed_bytes <- as.numeric(conn_log_malware_39$missed_bytes)
conn_log_malware_39[conn_log_malware_39=="(empty)"]<-"0"
conn_log_malware_39$tunnel_parents <- as.numeric(conn_log_malware_39$tunnel_parents)
#conn_log_malware_39[is.na(conn_log_malware_39)] <- 0
conn_log_malware_39$id.orig_p[is.na(conn_log_malware_39$id.orig_p)] <- 0
conn_log_malware_39$id.resp_p[is.na(conn_log_malware_39$id.resp_p)] <- 0
conn_log_malware_39$duration[is.na(conn_log_malware_39$duration)] <- 0
conn_log_malware_39$orig_bytes[is.na(conn_log_malware_39$orig_bytes)] <- 0
conn_log_malware_39$resp_bytes[is.na(conn_log_malware_39$resp_bytes)] <- 0
conn_log_malware_39$local_orig[is.na(conn_log_malware_39$local_orig)] <- 0
conn_log_malware_39$local_resp[is.na(conn_log_malware_39$local_resp)] <- 0
conn_log_malware_39$missed_bytes[is.na(conn_log_malware_39$missed_bytes)] <- 0
conn_log_malware_39$orig_pkts[is.na(conn_log_malware_39$orig_pkts)] <- 0
conn_log_malware_39$orig_ip_bytes[is.na(conn_log_malware_39$orig_ip_bytes)] <- 0
conn_log_malware_39$resp_pkts[is.na(conn_log_malware_39$resp_pkts)] <- 0
conn_log_malware_39$resp_ip_bytes[is.na(conn_log_malware_39$resp_ip_bytes)] <- 0
conn_log_malware_39$tunnel_parents[is.na(conn_log_malware_39$tunnel_parents)] <- 0
conn_log_malware_39$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_39)

###


###


conn_log_malware_42$timestamp <- NULL
conn_log_malware_42$userid <- NULL
```

```
conn_log_malware_42$id.orig_h <- NULL
conn_log_malware_42$id.resp_h <- NULL
conn_log_malware_42$duration <- as.numeric(conn_log_malware_42$duration)
conn_log_malware_42$orig_bytes <- as.numeric(conn_log_malware_42$orig_bytes)
conn_log_malware_42$resp_bytes <- as.numeric(conn_log_malware_42$resp_bytes)
conn_log_malware_42$resp_pkts <- as.numeric(conn_log_malware_42$resp_pkts)
conn_log_malware_42$orig_pkts <- as.numeric(conn_log_malware_42$orig_pkts)
conn_log_malware_42$orig_ip_bytes <- as.numeric(conn_log_malware_42$orig_ip_bytes)
conn_log_malware_42$resp_ip_bytes <- as.numeric(conn_log_malware_42$resp_ip_bytes)

conn_log_malware_42$service <- as.character(conn_log_malware_42$service)
conn_log_malware_42$local_orig <- as.numeric(conn_log_malware_42$local_orig)
conn_log_malware_42$local_resp <- as.numeric(conn_log_malware_42$local_resp)

#for character IVs

conn_log_malware_42[conn_log_malware_42=="-"]<-"0"


conn_log_malware_42$missed_bytes <- as.numeric(conn_log_malware_42$missed_bytes)
conn_log_malware_42[conn_log_malware_42=="(empty)"]<-"0"
conn_log_malware_42$tunnel_parents <- as.numeric(conn_log_malware_42$tunnel_parents)
#conn_log_malware_42[is.na(conn_log_malware_42)] <- 0
conn_log_malware_42$id.orig_p[is.na(conn_log_malware_42$id.orig_p)] <- 0
conn_log_malware_42$id.resp_p[is.na(conn_log_malware_42$id.resp_p)] <- 0
conn_log_malware_42$duration[is.na(conn_log_malware_42$duration)] <- 0
conn_log_malware_42$orig_bytes[is.na(conn_log_malware_42$orig_bytes)] <- 0
conn_log_malware_42$resp_bytes[is.na(conn_log_malware_42$resp_bytes)] <- 0
conn_log_malware_42$local_orig[is.na(conn_log_malware_42$local_orig)] <- 0
conn_log_malware_42$local_resp[is.na(conn_log_malware_42$local_resp)] <- 0
conn_log_malware_42$missed_bytes[is.na(conn_log_malware_42$missed_bytes)] <- 0
conn_log_malware_42$orig_pkts[is.na(conn_log_malware_42$orig_pkts)] <- 0
conn_log_malware_42$orig_ip_bytes[is.na(conn_log_malware_42$orig_ip_bytes)] <- 0
conn_log_malware_42$resp_pkts[is.na(conn_log_malware_42$resp_pkts)] <- 0
conn_log_malware_42$resp_ip_bytes[is.na(conn_log_malware_42$resp_ip_bytes)] <- 0
conn_log_malware_42$tunnel_parents[is.na(conn_log_malware_42$tunnel_parents)] <- 0
conn_log_malware_42$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_42)

###

conn_log_malware_43$timestamp <- NULL
conn_log_malware_43$userid <- NULL
```

```
conn_log_malware_43$id.orig_h <- NULL
conn_log_malware_43$id.resp_h <- NULL
conn_log_malware_43$duration <- as.numeric(conn_log_malware_43$duration)
conn_log_malware_43$orig_bytes <- as.numeric(conn_log_malware_43$orig_bytes)
conn_log_malware_43$resp_bytes <- as.numeric(conn_log_malware_43$resp_bytes)
conn_log_malware_43$resp_pkts <- as.numeric(conn_log_malware_43$resp_pkts)
conn_log_malware_43$orig_pkts <- as.numeric(conn_log_malware_43$orig_pkts)
conn_log_malware_43$orig_ip_bytes <- as.numeric(conn_log_malware_43$orig_ip_bytes)
conn_log_malware_43$resp_ip_bytes <- as.numeric(conn_log_malware_43$resp_ip_bytes)

conn_log_malware_43$service <- as.character(conn_log_malware_43$service)
conn_log_malware_43$local_orig <- as.numeric(conn_log_malware_43$local_orig)
conn_log_malware_43$local_resp <- as.numeric(conn_log_malware_43$local_resp)

#for character IVs

conn_log_malware_43[conn_log_malware_43=="-"]<-"0"


conn_log_malware_43$missed_bytes <- as.numeric(conn_log_malware_43$missed_bytes)
conn_log_malware_43[conn_log_malware_43=="(empty)"]<-"0"
conn_log_malware_43$tunnel_parents <- as.numeric(conn_log_malware_43$tunnel_parents)
#conn_log_malware_43[is.na(conn_log_malware_43)] <- 0
conn_log_malware_43$id.orig_p[is.na(conn_log_malware_43$id.orig_p)] <- 0
conn_log_malware_43$id.resp_p[is.na(conn_log_malware_43$id.resp_p)] <- 0
conn_log_malware_43$duration[is.na(conn_log_malware_43$duration)] <- 0
conn_log_malware_43$orig_bytes[is.na(conn_log_malware_43$orig_bytes)] <- 0
conn_log_malware_43$resp_bytes[is.na(conn_log_malware_43$resp_bytes)] <- 0
conn_log_malware_43$local_orig[is.na(conn_log_malware_43$local_orig)] <- 0
conn_log_malware_43$local_resp[is.na(conn_log_malware_43$local_resp)] <- 0
conn_log_malware_43$missed_bytes[is.na(conn_log_malware_43$missed_bytes)] <- 0
conn_log_malware_43$orig_pkts[is.na(conn_log_malware_43$orig_pkts)] <- 0
conn_log_malware_43$orig_ip_bytes[is.na(conn_log_malware_43$orig_ip_bytes)] <- 0
conn_log_malware_43$resp_pkts[is.na(conn_log_malware_43$resp_pkts)] <- 0
conn_log_malware_43$resp_ip_bytes[is.na(conn_log_malware_43$resp_ip_bytes)] <- 0
conn_log_malware_43$tunnel_parents[is.na(conn_log_malware_43$tunnel_parents)] <- 0
conn_log_malware_43$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_43)

###


###
```

```
conn_log_malware_44$timestamp <- NULL
conn_log_malware_44$userid <- NULL
conn_log_malware_44$id.orig_h <- NULL
conn_log_malware_44$id.resp_h <- NULL
conn_log_malware_44$duration <- as.numeric(conn_log_malware_44$duration)
conn_log_malware_44$orig_bytes <- as.numeric(conn_log_malware_44$orig_bytes)
conn_log_malware_44$resp_bytes <- as.numeric(conn_log_malware_44$resp_bytes)
conn_log_malware_44$resp_pkts <- as.numeric(conn_log_malware_44$resp_pkts)
conn_log_malware_44$orig_pkts <- as.numeric(conn_log_malware_44$orig_pkts)
conn_log_malware_44$orig_ip_bytes <- as.numeric(conn_log_malware_44$orig_ip_bytes)
conn_log_malware_44$resp_ip_bytes <- as.numeric(conn_log_malware_44$resp_ip_bytes)

conn_log_malware_44$service <- as.character(conn_log_malware_44$service)
conn_log_malware_44$local_orig <- as.numeric(conn_log_malware_44$local_orig)
conn_log_malware_44$local_resp <- as.numeric(conn_log_malware_44$local_resp)

#for character IVs

conn_log_malware_44[conn_log_malware_44=="-"]<-"0"


conn_log_malware_44$missed_bytes <- as.numeric(conn_log_malware_44$missed_bytes)
conn_log_malware_44[conn_log_malware_44=="(empty)"]<-"0"
conn_log_malware_44$tunnel_parents <- as.numeric(conn_log_malware_44$tunnel_parents)
#conn_log_malware_44[is.na(conn_log_malware_44)] <- 0
conn_log_malware_44$id.orig_p[is.na(conn_log_malware_44$id.orig_p)] <- 0
conn_log_malware_44$id.resp_p[is.na(conn_log_malware_44$id.resp_p)] <- 0
conn_log_malware_44$duration[is.na(conn_log_malware_44$duration)] <- 0
conn_log_malware_44$orig_bytes[is.na(conn_log_malware_44$orig_bytes)] <- 0
conn_log_malware_44$resp_bytes[is.na(conn_log_malware_44$resp_bytes)] <- 0
conn_log_malware_44$local_orig[is.na(conn_log_malware_44$local_orig)] <- 0
conn_log_malware_44$local_resp[is.na(conn_log_malware_44$local_resp)] <- 0
conn_log_malware_44$missed_bytes[is.na(conn_log_malware_44$missed_bytes)] <- 0
conn_log_malware_44$orig_pkts[is.na(conn_log_malware_44$orig_pkts)] <- 0
conn_log_malware_44$orig_ip_bytes[is.na(conn_log_malware_44$orig_ip_bytes)] <- 0
conn_log_malware_44$resp_pkts[is.na(conn_log_malware_44$resp_pkts)] <- 0
conn_log_malware_44$resp_ip_bytes[is.na(conn_log_malware_44$resp_ip_bytes)] <- 0
conn_log_malware_44$tunnel_parents[is.na(conn_log_malware_44$tunnel_parents)] <- 0
conn_log_malware_44$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_44)

###
```

```
###

conn_log_malware_48$timestamp <- NULL
conn_log_malware_48$userid <- NULL
conn_log_malware_48$id.orig_h <- NULL
conn_log_malware_48$id.resp_h <- NULL
conn_log_malware_48$duration <- as.numeric(conn_log_malware_48$duration)
conn_log_malware_48$orig_bytes <- as.numeric(conn_log_malware_48$orig_bytes)
conn_log_malware_48$resp_bytes <- as.numeric(conn_log_malware_48$resp_bytes)
conn_log_malware_48$resp_pkts <- as.numeric(conn_log_malware_48$resp_pkts)
conn_log_malware_48$orig_pkts <- as.numeric(conn_log_malware_48$orig_pkts)
conn_log_malware_48$orig_ip_bytes <- as.numeric(conn_log_malware_48$orig_ip_bytes)
conn_log_malware_48$resp_ip_bytes <- as.numeric(conn_log_malware_48$resp_ip_bytes)

conn_log_malware_48$service <- as.character(conn_log_malware_48$service)
conn_log_malware_48$local_orig <- as.numeric(conn_log_malware_48$local_orig)
conn_log_malware_48$local_resp <- as.numeric(conn_log_malware_48$local_resp)

#for character IVs

conn_log_malware_48[conn_log_malware_48=="-"]<-"0"


conn_log_malware_48$missed_bytes <- as.numeric(conn_log_malware_48$missed_bytes)
conn_log_malware_48[conn_log_malware_48=="(empty)"]<-"0"
conn_log_malware_48$tunnel_parents <- as.numeric(conn_log_malware_48$tunnel_parents)
#conn_log_malware_48[is.na(conn_log_malware_48)] <- 0
conn_log_malware_48$id.orig_p[is.na(conn_log_malware_48$id.orig_p)] <- 0
conn_log_malware_48$id.resp_p[is.na(conn_log_malware_48$id.resp_p)] <- 0
conn_log_malware_48$duration[is.na(conn_log_malware_48$duration)] <- 0
conn_log_malware_48$orig_bytes[is.na(conn_log_malware_48$orig_bytes)] <- 0
conn_log_malware_48$resp_bytes[is.na(conn_log_malware_48$resp_bytes)] <- 0
conn_log_malware_48$local_orig[is.na(conn_log_malware_48$local_orig)] <- 0
conn_log_malware_48$local_resp[is.na(conn_log_malware_48$local_resp)] <- 0
conn_log_malware_48$missed_bytes[is.na(conn_log_malware_48$missed_bytes)] <- 0
conn_log_malware_48$orig_pkts[is.na(conn_log_malware_48$orig_pkts)] <- 0
conn_log_malware_48$orig_ip_bytes[is.na(conn_log_malware_48$orig_ip_bytes)] <- 0
conn_log_malware_48$resp_pkts[is.na(conn_log_malware_48$resp_pkts)] <- 0
conn_log_malware_48$resp_ip_bytes[is.na(conn_log_malware_48$resp_ip_bytes)] <- 0
conn_log_malware_48$tunnel_parents[is.na(conn_log_malware_48$tunnel_parents)] <- 0
conn_log_malware_48$label <- NULL #redundant with the primary DV

#worked!
```

View(conn_log_malware_48)

###

###

```
conn_log_malware_49$timestamp <- NULL
conn_log_malware_49$userid <- NULL
conn_log_malware_49$id.orig_h <- NULL
conn_log_malware_49$id.resp_h <- NULL
conn_log_malware_49$duration <- as.numeric(conn_log_malware_49$duration)
conn_log_malware_49$orig_bytes <- as.numeric(conn_log_malware_49$orig_bytes)
conn_log_malware_49$resp_bytes <- as.numeric(conn_log_malware_49$resp_bytes)
conn_log_malware_49$resp_pkts <- as.numeric(conn_log_malware_49$resp_pkts)
conn_log_malware_49$orig_pkts <- as.numeric(conn_log_malware_49$orig_pkts)
conn_log_malware_49$orig_ip_bytes <- as.numeric(conn_log_malware_49$orig_ip_bytes)
conn_log_malware_49$resp_ip_bytes <- as.numeric(conn_log_malware_49$resp_ip_bytes)

conn_log_malware_49$service <- as.character(conn_log_malware_49$service)
conn_log_malware_49$local_orig <- as.numeric(conn_log_malware_49$local_orig)
conn_log_malware_49$local_resp <- as.numeric(conn_log_malware_49$local_resp)

#for character IVs

conn_log_malware_49[conn_log_malware_49=="-"]<-"0"


conn_log_malware_49$missed_bytes <- as.numeric(conn_log_malware_49$missed_bytes)
conn_log_malware_49[conn_log_malware_49=="(empty)"]<-"0"
conn_log_malware_49$tunnel_parents <- as.numeric(conn_log_malware_49$tunnel_parents)
#conn_log_malware_49[is.na(conn_log_malware_49)] <- 0
conn_log_malware_49$id.orig_p[is.na(conn_log_malware_49$id.orig_p)] <- 0
conn_log_malware_49$id.resp_p[is.na(conn_log_malware_49$id.resp_p)] <- 0
conn_log_malware_49$duration[is.na(conn_log_malware_49$duration)] <- 0
conn_log_malware_49$orig_bytes[is.na(conn_log_malware_49$orig_bytes)] <- 0
conn_log_malware_49$resp_bytes[is.na(conn_log_malware_49$resp_bytes)] <- 0
conn_log_malware_49$local_orig[is.na(conn_log_malware_49$local_orig)] <- 0
conn_log_malware_49$local_resp[is.na(conn_log_malware_49$local_resp)] <- 0
conn_log_malware_49$missed_bytes[is.na(conn_log_malware_49$missed_bytes)] <- 0
conn_log_malware_49$orig_pkts[is.na(conn_log_malware_49$orig_pkts)] <- 0
conn_log_malware_49$orig_ip_bytes[is.na(conn_log_malware_49$orig_ip_bytes)] <- 0
conn_log_malware_49$resp_pkts[is.na(conn_log_malware_49$resp_pkts)] <- 0
conn_log_malware_49$resp_ip_bytes[is.na(conn_log_malware_49$resp_ip_bytes)] <- 0
conn_log_malware_49$tunnel_parents[is.na(conn_log_malware_49$tunnel_parents)] <- 0
```

```
conn_log_malware_49$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_49)

###

###

conn_log_malware_52$timestamp <- NULL
conn_log_malware_52$userid <- NULL
conn_log_malware_52$id.orig_h <- NULL
conn_log_malware_52$id.resp_h <- NULL
conn_log_malware_52$duration <- as.numeric(conn_log_malware_52$duration)
conn_log_malware_52$orig_bytes <- as.numeric(conn_log_malware_52$orig_bytes)
conn_log_malware_52$resp_bytes <- as.numeric(conn_log_malware_52$resp_bytes)
conn_log_malware_52$resp_pkts <- as.numeric(conn_log_malware_52$resp_pkts)
conn_log_malware_52$orig_pkts <- as.numeric(conn_log_malware_52$orig_pkts)
conn_log_malware_52$orig_ip_bytes <- as.numeric(conn_log_malware_52$orig_ip_bytes)
conn_log_malware_52$resp_ip_bytes <- as.numeric(conn_log_malware_52$resp_ip_bytes)

conn_log_malware_52$service <- as.character(conn_log_malware_52$service)
conn_log_malware_52$local_orig <- as.numeric(conn_log_malware_52$local_orig)
conn_log_malware_52$local_resp <- as.numeric(conn_log_malware_52$local_resp)

#for character IVs

conn_log_malware_52[conn_log_malware_52=="-"]<-"0"


conn_log_malware_52$missed_bytes <- as.numeric(conn_log_malware_52$missed_bytes)
conn_log_malware_52[conn_log_malware_52=="(empty)"]<-"0"
conn_log_malware_52$tunnel_parents <- as.numeric(conn_log_malware_52$tunnel_parents)
#conn_log_malware_52[is.na(conn_log_malware_52)] <- 0
conn_log_malware_52$id.orig_p[is.na(conn_log_malware_52$id.orig_p)] <- 0
conn_log_malware_52$id.resp_p[is.na(conn_log_malware_52$id.resp_p)] <- 0
conn_log_malware_52$duration[is.na(conn_log_malware_52$duration)] <- 0
conn_log_malware_52$orig_bytes[is.na(conn_log_malware_52$orig_bytes)] <- 0
conn_log_malware_52$resp_bytes[is.na(conn_log_malware_52$resp_bytes)] <- 0
conn_log_malware_52$local_orig[is.na(conn_log_malware_52$local_orig)] <- 0
conn_log_malware_52$local_resp[is.na(conn_log_malware_52$local_resp)] <- 0
conn_log_malware_52$missed_bytes[is.na(conn_log_malware_52$missed_bytes)] <- 0
conn_log_malware_52$orig_pkts[is.na(conn_log_malware_52$orig_pkts)] <- 0
conn_log_malware_52$orig_ip_bytes[is.na(conn_log_malware_52$orig_ip_bytes)] <- 0
conn_log_malware_52$resp_pkts[is.na(conn_log_malware_52$resp_pkts)] <- 0
```

```
conn_log_malware_52$resp_ip_bytes[is.na(conn_log_malware_52$resp_ip_bytes)] <- 0
conn_log_malware_52$tunnel_parents[is.na(conn_log_malware_52$tunnel_parents)] <- 0
conn_log_malware_52$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_52)

###

###

conn_log_malware_60$timestamp <- NULL
conn_log_malware_60$userid <- NULL
conn_log_malware_60$id.orig_h <- NULL
conn_log_malware_60$id.resp_h <- NULL
conn_log_malware_60$duration <- as.numeric(conn_log_malware_60$duration)
conn_log_malware_60$orig_bytes <- as.numeric(conn_log_malware_60$orig_bytes)
conn_log_malware_60$resp_bytes <- as.numeric(conn_log_malware_60$resp_bytes)
conn_log_malware_60$resp_pkts <- as.numeric(conn_log_malware_60$resp_pkts)
conn_log_malware_60$orig_pkts <- as.numeric(conn_log_malware_60$orig_pkts)
conn_log_malware_60$orig_ip_bytes <- as.numeric(conn_log_malware_60$orig_ip_bytes)
conn_log_malware_60$resp_ip_bytes <- as.numeric(conn_log_malware_60$resp_ip_bytes)

conn_log_malware_60$service <- as.character(conn_log_malware_60$service)
conn_log_malware_60$local_orig <- as.numeric(conn_log_malware_60$local_orig)
conn_log_malware_60$local_resp <- as.numeric(conn_log_malware_60$local_resp)

#for character IVs

conn_log_malware_60[conn_log_malware_60=="-"]<-"0"


conn_log_malware_60$missed_bytes <- as.numeric(conn_log_malware_60$missed_bytes)
conn_log_malware_60[conn_log_malware_60=="(empty)"]<-"0"
conn_log_malware_60$tunnel_parents <- as.numeric(conn_log_malware_60$tunnel_parents)
#conn_log_malware_60[is.na(conn_log_malware_60)] <- 0
conn_log_malware_60$id.orig_p[is.na(conn_log_malware_60$id.orig_p)] <- 0
conn_log_malware_60$id.resp_p[is.na(conn_log_malware_60$id.resp_p)] <- 0
conn_log_malware_60$duration[is.na(conn_log_malware_60$duration)] <- 0
conn_log_malware_60$orig_bytes[is.na(conn_log_malware_60$orig_bytes)] <- 0
conn_log_malware_60$resp_bytes[is.na(conn_log_malware_60$resp_bytes)] <- 0
conn_log_malware_60$local_orig[is.na(conn_log_malware_60$local_orig)] <- 0
conn_log_malware_60$local_resp[is.na(conn_log_malware_60$local_resp)] <- 0
conn_log_malware_60$missed_bytes[is.na(conn_log_malware_60$missed_bytes)] <- 0
conn_log_malware_60$orig_pkts[is.na(conn_log_malware_60$orig_pkts)] <- 0
```

```
conn_log_malware_60$orig_ip_bytes[is.na(conn_log_malware_60$orig_ip_bytes)] <- 0
conn_log_malware_60$resp_pkts[is.na(conn_log_malware_60$resp_pkts)] <- 0
conn_log_malware_60$resp_ip_bytes[is.na(conn_log_malware_60$resp_ip_bytes)] <- 0
conn_log_malware_60$tunnel_parents[is.na(conn_log_malware_60$tunnel_parents)] <- 0
conn_log_malware_60$label <- NULL #redundant with the primary DV

#worked!

View(conn_log_malware_60)

###


#now start combining these sub-dataframes into the single final "master" dataframe using dplyr's bind_row function.

#WIP v1..

#perform memory garbage collection after all of this large dataframe manipulation, to free up any memory that is still in the cache.
gc()


final_master_df <- bind_rows(conn_log_malware_1, conn_log_malware_3, conn_log_malware_7, conn_log_malware_8, conn_log_malware_9, conn_log_malware_17, conn_log_malware_20, conn_log_malware_21, conn_log_malware_33, conn_log_malware_34, conn_log_malware_35, conn_log_malware_36, conn_log_malware_39, conn_log_malware_42, conn_log_malware_43, conn_log_malware_44, conn_log_malware_48, conn_log_malware_49, conn_log_malware_52, conn_log_malware_60)

#setwd("C://ISU_PhD//COT711_Dissertation//")
#write.csv(final_master_df, "final_master_df.csv")

#finished here on evening of 9/5/22. Master DF is fully merged. Still need to balance these factor classes with raw row copies:
# C&C, C&C-Torii, C&C-HeartBeat, C&C-Mirai, C&C-HeartBeat-Attack, C&C-PartOfAHorizontalPortScan, Attack, C&C-FileDownload, Okiru-Attack, Pa
################################################################################
################################################################################

setwd("C://ISU_PhD//COT711_Dissertation//")
getwd()
#[1] "C:/ISU_PhD/COT711_Dissertation"

 library(readr)
 final_master_df <- read_csv("final_master_df.csv")
```

```
#remove artifact first IV (row count IV) created by the write.csv function to the output df.
final_master_df$..1 <- NULL



#New names:
#• `` -> `..1`
#Rows: 146924220 Columns: 19
#── Column specification ────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────
─────
#Delimiter: ","
#chr (5): proto, service, conn_state, history, detailed-label
#dbl (14): ..1, id.orig_p, id.resp_p, duration, orig_bytes, resp_bytes, local_orig, local..
#
#ℹ Use `spec()` to retrieve the full column specification for this data.
#ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.

 View(final_master_df)


#free up junk from RAM with garbage collection routine.
gc()
# used (Mb) gc trigger (Mb) max used (Mb)
#Ncells 863013 46.1 1479600 79.1 1479600 79.1
#Vcells 2718659811 20741.8 5418587411 41340.6 5363517157 40920.4

#show the class balance current state
 table(final_master_df$`detailed-label`)

# - 0
# 75955 11139745
# Attack C&C
# 8865 17613
# C&C-FileDownload C&C-HeartBeat
# 53 26962
# C&C-HeartBeat-Attack C&C-HeartBeat-FileDownload
# 834 11
# C&C-Mirai C&C-PartOfAHorizontalPortScan
# 2 888
# C&C-Torii DDoS
# 30 12267750
# FileDownload Okiru
# 18 38317947
```

```
# Okiru-Attack Pa
# 3 1
# PartOfAHorizontalPortScan PartOfAHorizontalPortScan-Attack
# 85067519 5

 #manually balance these ultra-minority classes to ~500k each

#for research paper one, convert all of these character IVs to factors, then convert those factors (nominal data) into
#numeric values (e.g., recode the nominal qualitative data into ordinal numeric dummy variables, then run an npmanova/permanova test.

#then write up the results, hopefully with a p-value below an alpha test statistic so we can reject our null hypothesis.


final_master_df$proto <- as.factor(final_master_df$proto)
final_master_df$service <- as.factor(final_master_df$service)
final_master_df$conn_state <- as.factor(final_master_df$conn_state)
final_master_df$history <- as.factor(final_master_df$history)
final_master_df$`detailed-label` <- as.factor(final_master_df$`detailed-label`) #convert the DV to factors

#now convert these nominal factors to ordinal values (numeric scores) by recoding them to as.numeric, which uses their sorted index values

final_master_df$proto <- as.numeric(final_master_df$proto)
final_master_df$service <- as.numeric(final_master_df$service)
final_master_df$conn_state <- as.numeric(final_master_df$conn_state)
final_master_df$history <- as.numeric(final_master_df$history)
final_master_df$`detailed-label` <- as.numeric(final_master_df$`detailed-label`) #convert the DV to ordinal

#bring vegan package into memory
library(vegan)

y <- final_master_df
y$`detailed.label` <- NULL
#y becomes my predictor variables list (IVs), without the DV present.


#x1 <- adonis(y ~ final_master_df$`detailed-label`, method = "euclidean", permutations = 999)

#I need to randomly subsample the df in order to be able to be processed by adonis.
library(dplyr)
```

```
set.seed(123)
#subset this very large dataframe by randomly selecting 75000 rows of data, to be able to fit the permanova function vector into RAM
final_master_df_small <- sample_n(final_master_df, 75000)

#also do 70k subsample (easier), and still meets critieria from the Chi-square goodness of fit sample size test

y2 <- final_master_df_small
y2$`detailed-label` <- NULL

x2 <- adonis(y2 ~ final_master_df_small$`detailed-label`, method = "euclidean", permutations = 200) #alpha = 0.05


#now flip the y and x variables around, so I can see which specific IVs are statistically significant against our DV.
y2 <- final_master_df_small
y2$`detailed-label` <- NULL

final_master_df_small$..1 <- NULL

 x1 <- colnames(final_master_df_small)[colnames(final_master_df_small) != "detailed-label"]
 x1

#make the 'detailed-label' the true DV for a predictive model, and get the individual IVs that are statistically significant, along
#with the residuals for the entire model so I can show both the p-value for the entire model and the p-values for the best KRIs.
x2b <- adonis2(final_master_df_small$'detailed-label' ~ x1, method = "euclidean", permutations = 200) #alpha = 0.05
x2b


#see email notes re: Chi-Squared Goodness of Fit test to determine random sample size for 20 groups with the effect size parameters I've selected.


#see https://www.researchgate.net/topic/Adonis, adonis function will work with unbalanced design group data.
#however I may want to use method = "bray" if Euclidean becomes problematic for me.

#x4$aov.tab
#Permutation: free
#Number of permutations: 200
#
```

```
#Terms added sequentially (first to last)
#
# Df SumsOfSqs MeanSqs F.Model R2 Pr(>F)
#final_master_df_small$`detailed-label` 1 2.0887e+18 2.0887e+18 4.9588 0.00017 0.02985 *
#Residuals 29998 1.2635e+22 4.2121e+17 0.99983
#Total 29999 1.2637e+22 1.00000
#---
#Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

x4$coefficients #use a function to format this "more nicer" for publication :)

# id.orig_p id.resp_p proto service
#(Intercept) 32598.8016 24731.1250 2.038494345 1.882805319
#final_master_df_small$`detailed-label` 274.1975 -442.1074 -0.002402864 0.002810256
# duration orig_bytes resp_bytes conn_state
#(Intercept) 1.37163018 42990743 2.1899780 6.329912
#final_master_df_small$`detailed-label` -0.07898488 -2065813 -0.1347239 0.032525
# local_orig local_resp missed_bytes history
#(Intercept) 0 0 0 62.7266961
#final_master_df_small$`detailed-label` 0 0 0 0.2737409
# orig_pkts orig_ip_bytes resp_pkts resp_ip_bytes
#(Intercept) 1.648871362 98.191130 0.022566426 3.2372304
#final_master_df_small$`detailed-label` -0.007264689 -2.139809 -0.001313123 -0.1959011
# tunnel_parents
#(Intercept) 0
#final_master_df_small$`detailed-label` 0

#we are statistically significant against the alpha test statistic :)



#permutations 999 is equivalent of stating that my alpha test statistic is 0.01 (1 out of 999 events taking place due to random chance)



#####
#start the h2o model build process here.
library(h2o)
h2o.init(nthreads=-1, max_mem_size = "50G")


#factorize / convert the original raw df before bringing it into h2o.
final_master_df$proto <- as.factor(final_master_df$proto)
```

```
final_master_df$service <- as.factor(final_master_df$service)
final_master_df$conn_state <- as.factor(final_master_df$conn_state)
final_master_df$history <- as.factor(final_master_df$history)
final_master_df$`detailed-label` <- as.factor(final_master_df$`detailed-label`) #convert the DV to factors

#now convert these nominal factors to ordinal values (numeric scores) by recoding them to as.numeric, which uses their sorted index values

#final_master_df$proto <- as.numeric(final_master_df$proto)
#final_master_df$service <- as.numeric(final_master_df$service)
#final_master_df$conn_state <- as.numeric(final_master_df$conn_state)
#final_master_df$history <- as.numeric(final_master_df$history)
#final_master_df$`detailed-label` <- as.numeric(final_master_df$`detailed-label`) #convert the DV to ordinal



#iris.h2o <- as.h2o(localH2O, iris.r, key="iris.h2o")
final_master_df.h2o <- as.h2o(final_master_df, key="final_master_df.h2o")

#clean out the original raw df so I free up memory.
final_master_df <- NULL

##################################now build an ANN classification predictive model.

set.seed(123)
# Split dataset giving the training dataset 70% of the data
final_master_split <- h2o.splitFrame(data = final_master_df.h2o, ratios = 0.70)
print(dim(final_master_split[[1]]))
#[1] 102847073 18
print(dim(final_master_split[[2]]))
#[1] 44077147 18

# Create a training set from the 1st dataset in the split
final_master_train <- final_master_split[[1]]

# Create a testing set from the 2nd dataset in the split
final_master_test <- final_master_split[[2]]

#build the working prototype ANN model for RQ1 :)

#############################################################################
# Build and train the model:
dl <- h2o.deeplearning(x = 1:17,
 y = "detailed-label",
```

```
distribution = "multinomial",
hidden = c(40,40,40),
epochs = 10,
overwrite_with_best_model = T,
reproducible = FALSE,
activation = "Tanh",
single_node_mode = FALSE,
balance_classes = TRUE,
force_load_balance = FALSE,
seed = 12345,
categorical_encoding = "Eigen",
rate = 0.05,
adaptive_rate = T,
score_training_samples = 10000,
score_validation_samples = 10000,
training_frame = final_master_train,
validation_frame = final_master_test,
variable_importances = T,
stopping_metric = "misclassification",
stopping_tolerance = 0.01,
stopping_rounds = 0 )


#see: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html




#continue here on tues. 9/6/22, modify this code to apply to each of the special minority classes listed above, but apply it to the
#final_master_df dataframe, not the individual sub dataframes.

#ZZZZZ

library(tidyverse)

conn_log_malware_1 %>% filter(`detailed-label` == "C&C")
conn_log_malware_1_C&C <- conn_log_malware_1 %>% filter(`detailed-label` == "C&C")
conn_log_malware_1_CC <- conn_log_malware_1 %>% filter(`detailed-label` == "C&C")
View(conn_log_malware_1_CC)
#worked!!!

#now upsample conn_log_malware_1_CC by 66200 times.
```

```
rebalanced_conn_log_malware_1_CC <-
conn_log_malware_1_CC[rep(seq_len(nrow(conn_log_malware_1_CC)), each = 66200), ]
View(rebalanced_conn_log_malware_1_CC)

#now rbind this rebalanced df into the conn_log_malware_1 df
conn_log_malware_1_balanced <- bind_rows(conn_log_malware_1,
rebalanced_conn_log_malware_1_CC)

table(conn_log_malware_1$`detailed-label`)

#before:
#0 C&C PartOfAHorizontalPortScan
#469275 8 539465


table(conn_log_malware_1_balanced$`detailed-label`)

#after:
#0 C&C PartOfAHorizontalPortScan
#469275 529608 539465



table(conn_log_malware_1$`detailed-label`)

#before:
#0 C&C PartOfAHorizontalPortScan
#469275 8 539465


table(conn_log_malware_1_balanced$`detailed-label`)

#after:
#0 C&C PartOfAHorizontalPortScan
#469275 529608 539465


#once that takes place then convert the master df character IVs to factors, including the DV.
Then start creating the first h2o model for RQ1.



#now convert the character IVs above to factors, but since we are factorizing this nominal data,
we don't want to convert the IVs to
#factors until the very end when all dataframe segments are merged together. So that becomes
the last step before we randomly subset
```

```
#the data into the new training dataset.



#in my Results section, discuss how I manually oversampled the C&C class (code above), and
will also probably do the same with the
#C&C-Torii, C&C-HeartBeat, C&C-Mirai, C&C-HeartBeat-Attack, C&C-
PartOfAHorizontalPortScan, Attack, C&C-FileDownload,
#Okiru-Attack, Pa classes which were also each very rare (~30 observations or less).
#All other classes can be balanced inside h2o using statistical methods
#inside the h2o cluster as sufficient raw observations exist to provide baseline pattern data.

#for the statistical class balancing, this will all be handled inside the h2o cluster model as a
parameter for the deep learning ANN.



#View DV classes from the first half of all dataframes.

 table(conn_log$`detailed-label`)

# -
#452

 table(conn_log_5_1$`detailed-label`)

# -
#1374

 table(conn_log_7_1$`detailed-label`)

# -
#130

 table(conn_log_malware_1$`detailed-label`)

# - C&C PartOfAHorizontalPortScan
# 469275 8 539465

 table(conn_log_malware_3$`detailed-label`)

# - Attack C&C
# 4536 5962 8
```

```
#PartOfAHorizontalPortScan
# 145597

 table(conn_log_malware_7$`detailed-label`)

# - C&C-HeartBeat DDoS Okiru
# 75955 5778 39584 11333397

 table(conn_log_malware_8$`detailed-label`)

# - C&C
#2181 8222

 table(conn_log_malware_9$`detailed-label`)

# - PartOfAHorizontalPortScan
# 22548 6355745

 table(conn_log_malware_17$`detailed-label`)

# - Attack
# 14567 4
# C&C-HeartBeat DDoS
# 3198 6450012
# Okiru PartOfAHorizontalPortScan
# 6449728 12899552
#PartOfAHorizontalPortScan-Attack
# 5

 table(conn_log_malware_20$`detailed-label`)

# - C&C-Torii
# 3193 16

table(conn_log_malware_21$`detailed-label`)

# - C&C-Torii
# 3272 14



table(conn_log_malware_33$`detailed-label`)


# - C&C-HeartBeat Okiru
# 629104 2203 6180343
```

#PartOfAHorizontalPortScan
17917351

table(conn_log_malware_34$`detailed-label`)

- C&C DDoS
1923 6706 14394
#PartOfAHorizontalPortScan
122

table(conn_log_malware_35$`detailed-label`)

- Attack C&C C&C-FileDownload DDoS
8262389 3 81 12 2185302
#

table(conn_log_malware_36$`detailed-label`)

- C&C-HeartBeat Okiru Okiru-Attack
#2663 15688 13626744 3

table(conn_log_malware_39$`detailed-label`)

- Attack C&C
1514 144 333
Pa PartOfAHorizontalPortScan
1 15511726

table(conn_log_malware_42$`detailed-label`)

#FileDownload
3

table(conn_log_malware_43$`detailed-label`)

- C&C C&C-FileDownload
1706280 313 14
FileDownload Okiru PartOfAHorizontalPortScan
1 727735 3127319

```
table(conn_log_malware_44$`detailed-label`)

# - C&C C&C-FileDownload DDoS
# 211 14 11 1
#



table(conn_log_malware_48$`detailed-label`)

# - Attack C&C-HeartBeat-Attack
# 3734 2752 834
# C&C-HeartBeat-FileDownload C&C-PartOfAHorizontalPortScan PartOfAHorizontalPortScan
# 11 888 3386119


 table(conn_log_malware_49$`detailed-label`)

# - C&C C&C-FileDownload
# 3665 1922 1
# FileDownload PartOfAHorizontalPortScan
# 14 5404959


table(conn_log_malware_52$`detailed-label`)

# - C&C C&C-FileDownload
# 1794 6 12
# C&C-Mirai PartOfAHorizontalPortScan
# 2 19779564


table(conn_log_malware_60$`detailed-label`)

# - C&C-HeartBeat DDoS
# 2476 95 3578457



#after all of these dataframes get trimmed (kill redundant DV and the biasing IVs) I can use the dplyr package and bind_rows
#to merge all 23 of these dataframes into a single dataset, THEN utilize another package to randomly subsample ~500k observations
```

```
#from each one, and THEN save that off as the "final" dataframe to be used for all 4 RQs, and
create an 80/20 train:test split of this
#subsampled data.

#Then the data munging/data prep is all done, and it's just building the models and permanova
validations, etc. etc. and writing
#up all of the results in the paper.


getwd()
#[1] "C:/ISU_PhD/COT711_Dissertation"

 # save the model
 model_path <- h2o.saveModel(object = dl, path = getwd(), force = TRUE)
 print(model_path)
#[1] "C:\\ISU_PhD\\COT711_Dissertation\\DeepLearning_model_R_1663034499858_1"

#see: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/save-and-load-model.html for loading/saving
h2o DL models in R.

#also see second note file, dl_ANN_h2o_validation_metrics_15sep2022.txt for more extensive
validation and dl model performance evaluation metrics including RMSE against the
#validation partition and the KRIs, etc.


 h2o.varimp_plot(dl)

 h2o.varimp(dl)

#Variable Importances:
# variable relative_importance scaled_importance percentage
#1 orig_ip_bytes 1.000000 1.000000 0.343957
#2 orig_pkts 0.343250 0.343250 0.118064
#3 missed_bytes 0.118205 0.118205 0.040657
#4 id.resp_p 0.089839 0.089839 0.030901
#5 history.S 0.072771 0.072771 0.025030
#
#---
# variable relative_importance scaled_importance percentage
#381 history.ShAdDaFRRRf 0.000505 0.000505 0.000174
#382 history.ShADFfr 0.000504 0.000504 0.000173
#383 history.missing(NA) 0.000000 0.000000 0.000000
#384 conn_state.missing(NA) 0.000000 0.000000 0.000000
#385 service.missing(NA) 0.000000 0.000000 0.000000
#386 proto.missing(NA) 0.000000 0.000000 0.000000
```

```
 varimpdetails <- h2o.varimp(dl)
 View(varimpdetails)
 write.csv(varimpdetails, "dl_varimpdetails_KRIs_15sep2022.csv")

#see varimpdetails .csv export file for the full analysis of KRIs. Note that these KRIs are the
result of one-hot encoding of all of the IVs. Would have been faster to use
#Eigen factor encoding but it's OK for now.


#now predict on the h2o test partition of the dataframe:

pred2 <- h2o.predict(dl, final_master_test)


################################################################################
##########################

#work on the permanova test for the result of the ANN model (dl) against a random subset of the
test partition (use the 19th column as the DV, e.g. my ANN prediction output).
#need to determine what the random sample size is, and then run that random sample against
adonis (will take awhile).
#This will finalize RQ1.

################################################################################
################################

#code from 9/18 to do this final before and after permanova on full model (using subset data
with 40k random observations) to test for whole-model statistical significance.

converted_test_with_ANN_predicted_15sep2022 <-
read.csv("C:/ISU_PhD/COT711_Dissertation/converted_test_with_ANN_predicted_15sep2022.
csv")

View(converted_test_with_ANN_predicted_15sep2022)
converted_test_with_ANN_predicted_15sep2022$X <- NULL

View(converted_test_with_ANN_predicted_15sep2022)

 converted_test_with_ANN_predicted_15sep2022$proto <-
as.factor(converted_test_with_ANN_predicted_15sep2022$proto)
 converted_test_with_ANN_predicted_15sep2022$service <-
as.factor(converted_test_with_ANN_predicted_15sep2022$service)
 converted_test_with_ANN_predicted_15sep2022$conn_state <-
as.factor(converted_test_with_ANN_predicted_15sep2022$conn_state)
 converted_test_with_ANN_predicted_15sep2022$history <-
as.factor(converted_test_with_ANN_predicted_15sep2022$history)
```

```
 converted_test_with_ANN_predicted_15sep2022$detailed.label <-
as.factor(converted_test_with_ANN_predicted_15sep2022$detailed.label)
 converted_test_with_ANN_predicted_15sep2022$dl_prediction <-
as.factor(converted_test_with_ANN_predicted_15sep2022$dl_prediction)

 converted_test_with_ANN_predicted_15sep2022$proto <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$proto)
 converted_test_with_ANN_predicted_15sep2022$service <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$service)
 converted_test_with_ANN_predicted_15sep2022$conn_state <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$conn_state)
 converted_test_with_ANN_predicted_15sep2022$history <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$history)
 converted_test_with_ANN_predicted_15sep2022$detailed.label <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$detailed.label)

#need to convert the DV to numeric as well; vegan requires all data to be numeric. I can decode
the DV numeric values later by creating a copy of the original column.
 converted_test_with_ANN_predicted_15sep2022$dl_prediction <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$dl_prediction)

 after <- converted_test_with_ANN_predicted_15sep2022
 before <- converted_test_with_ANN_predicted_15sep2022
 before$dl_prediction <- NULL
 after$detailed.label <- NULL

library(dplyr)

#40k subsample (easier), and still meets critieria from the Chi-square goodness of fit sample size
test for statistical power with 23 categories
set.seed(123)
#subset this very large dataframe by randomly selecting 40000 rows of data, to be able to fit the
permanova function vector into RAM
before_small <- sample_n(before, 40000)
x

#temporarily do subset to 1k, just to test performance of the function.
set.seed(123)
#subset this very large dataframe by randomly selecting 40000 rows of data, to be able to fit the
permanova function vector into RAM
before_small <- sample_n(before, 18000)
after_small <- sample_n(after, 18000)


y2 <- before_small
y2$`detailed-label` <- NULL
```

```
#x2 <- adonis(y2 ~ final_master_df_small$`detailed-label`, method = "euclidean", permutations = 200) #alpha = 0.05
#try method = euclidean OR method = bray, to see which works better.
x2 <- adonis2(y2 ~ before_small$detailed.label, method = "bray", permutations = 200, by=NULL) #alpha = 0.005, with by=NULL meaning model as a whole regardless of IV ordering is tested for significance.


x2$`Pr(>F)`
#[1] 0.004975124 NA NA
x2

#Permutation test for adonis under reduced model
#Permutation: free
#Number of permutations: 200
#
#adonis2(formula = y2 ~ before_small$detailed.label, permutations = 200, method = "bray", by = NULL)
# Df SumOfSqs R2 F Pr(>F)
#Model 1 16.1 0.00363 145.79 0.004975 **
#Residual 39998 4420.8 0.99637
#Total 39999 4436.9 1.00000
#---
#Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

x2$Df
#[1] 1 39998 39999
x2$R2
#[1] 0.003631779 0.996368221 1.000000000
x2$F
#[1] 145.7934 NA NA
x2$`Pr(>F)`
#[1] 0.004975124 NA NA

#now run the model for the "after" results from the ANN predictive model.

after_small <- sample_n(after, 40000)

after_small$dl_prediction <- as.factor(after_small$dl_prediction)
after_small$dl_prediction <- as.numeric(after_small$dl_prediction)

y2 <- NULL

y2 <- after_small
y2$dl_prediction <- NULL
```

```
x2 <- adonis2(y2 ~ after_small$dl_prediction, method = "bray", permutations = 200, by=NULL) #alpha = 0.005, with by=NULL meaning model as a whole regardless of IV ordering is tested for significance.

#note: 44m test partition observations from IoT-23 (30% total population) were reduced to 40,000 randomly sampled observations for use by the 200 permutation PERMANOVA test model.



x2 <- adonis2(after_small$dl_prediction ~ y2$id.orig_p+y2$id.resp_p+y2$proto+y2$service+y2$duration+y2$orig_bytes+y2$resp_bytes+y2$conn_state+y2$local_orig+y2$local_resp+y2$missed_bytes+y2$history+y2$orig_pkts+y2$orig_ip_bytes+y2$resp_pkts+y2$resp_ip_bytes+y2$tunnel_parents, method = "bray", permutations = 20, by=NULL) #alpha = 0.005, with by=NULL meaning model as a whole regardless of IV ordering is tested for significance.
x2

#permanova output of my ANN predictive test partition (randomly subsampled per Chi-square Goodness of Fit test):

Permutation test for adonis under reduced model
Permutation: free
Number of permutations: 20

adonis2(formula = after_small$dl_prediction ~ y2$id.orig_p + y2$id.resp_p + y2$proto + y2$service + y2$duration + y2$orig_bytes + y2$resp_bytes + y2$conn_state + y2$local_orig + y2$local_resp + y2$missed_bytes + y2$history + y2$orig_pkts + y2$orig_ip_bytes + y2$resp_pkts + y2$resp_ip_bytes + y2$tunnel_parents, permutations = 20, method = "bray", by = NULL)
 Df SumOfSqs R2 F Pr(>F)
Model 13 173.99 0.25057 1028.4 0.04762 *
Residual 39986 520.40 0.74943
Total 39999 694.39 1.00000
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

#now calculate the beta dispersal, based on https://chrischizinski.github.io/rstats/adonis/

library(vegan)

## Bray-Curtis distances between samples
#dis <- vegdist(all.sites)
dis <- vegdist(after_small) #kicked off at 10:30am Fri 9/23/22..
```

```
#may need to call betadisrp with a smaller subset than 40k for after_small..

#next do:
## Calculate multivariate dispersions
#mod <- betadisper(dis, trt)
mod <- betadisper(dis, after_small$dl_prediction)
mod

#alternatively, manually grab n number of samples from each group in the original (post-
forecasted) df, so all ~18 class types are represented.
#use those n number of observations from each group for the betadispr function.

# mod

#         Homogeneity of multivariate dispersions
#
#Call: betadisper(d = dis, group = after_small$dl_prediction)
#
#No. of Positive Eigenvalues: 88
#No. of Negative Eigenvalues: 151
#
#Average distance to median:
# 2 6 11 13 15
#0.1079 0.0000 0.1585 0.0972 0.2081
#
#Eigenvalues for PCoA axes:
#(Showing 8 of 239 eigenvalues)
# PCoA1 PCoA2 PCoA3 PCoA4 PCoA5 PCoA6 PCoA7 PCoA8
#82.8953 25.5788 6.2673 4.5102 2.8503 1.6121 1.2032 0.9142
#record analytical output results

> mod <- betadisper(dis, after_small$dl_prediction)
Warning message:
In betadisper(dis, after_small$dl_prediction) :
 some squared distances are negative and changed to zero
> mod

        Homogeneity of multivariate dispersions

Call: betadisper(d = dis, group = after_small$dl_prediction)

No. of Positive Eigenvalues: 468
No. of Negative Eigenvalues: 898

Average distance to median:
 1 2 4 6 11 13 15
```

```
0.29945 0.09943 0.04429 0.01938 0.18365 0.10657 0.21675

Eigenvalues for PCoA axes:
(Showing 8 of 1366 eigenvalues)
 PCoA1 PCoA2 PCoA3 PCoA4 PCoA5 PCoA6 PCoA7 PCoA8
1528.06 510.61 121.30 83.20 57.70 29.84 19.12 13.39
>
> Sys.time()
[1] "2022-09-27 11:30:48 EDT"



#also try:

plot(mod)

boxplot(mod)

#and also see visualizing the multivariate homogeneity of group dispersions using methods
shown in:
#https://chrischizinski.github.io/rstats/adonis/

#re-run this betadisp procedure but manually grab n number of rows from each group
hist(table(after_small$dl_prediction))
hist(table(after$dl_prediction))
hist(table(before$detailed.label))

#use dplyr function:
new_df <- df %>% group_by(ID) %>% slice_sample(n=500) # do this for each group within the
DV, into a separate sub-dataframe.
#then merge all of these x sub-dataframes into a new "after" master dataframe, with which to
perform the betadispr test on.
then also try:
#plot(mod)

#boxplot(mod)


new_df <- converted_test_with_ANN_predicted_15sep2022 %>%
group_by(dl_prediction) %>% slice_sample(n=500, replace = TRUE)
#replace = true tells dplyr to repeat group elements when n < 500 observastions
View(new_df)
#works! conduct the betadispr test on new_df
```

```
#recode the factor IVs to ordinal numeric data so it can be processed properly. Note that this is
the random-sample ANN predictive model, e.g. "after" version.
 new_df$proto <- as.factor(new_df$proto)
 new_df$service <- as.factor(new_df$service)
 new_df$conn_state <- as.factor(new_df$conn_state)
 new_df$history <- as.factor(new_df$history)
 new_df$detailed.label <- as.factor(new_df$detailed.label)
 new_df$dl_prediction <- as.factor(new_df$dl_prediction)

 new_df$proto <- as.numeric(new_df$proto)
 new_df$service <- as.numeric(new_df$service)
 new_df$conn_state <- as.numeric(new_df$conn_state)
 new_df$history <- as.numeric(new_df$history)
 new_df$detailed.label <- as.numeric(new_df$detailed.label)

 new_df$dl_prediction <- as.numeric(new_df$dl_prediction)

#get rid of the original DV first!
new_df$detailed.label <- NULL

 dis <- vegdist(new_df) #kicked off at 11:10pm Wed 9/25/22..
 mod <- betadisper(dis, new_df$dl_prediction)
#took about 20 minutes with 8000 rows of data (19 classes x 500 observations each)

#create plots (plot and boxplot) of this new_df beta dispersion, which is for the "after" random
sample (e.g. the ANN predictive model). Be use to use main= parameter in the plot().

#repeat this process for the before random sample using sample_n, and produce the same 2 plots.
Record the betadispr coefficients scores. Then good to go to finalize paper AND rq1 section of
the dissertation. Move on to RQ2.

mod # show results for the sub-sampled but balanced "after" ANN predictive output:


        Homogeneity of multivariate dispersions

Call: betadisper(d = dis, group = new_df$dl_prediction)

No. of Positive Eigenvalues: 765
No. of Negative Eigenvalues: 1546

Average distance to median:
 1 2 3 4 5 6 7 8 9
3.406e-01 9.566e-02 1.370e-01 4.635e-02 1.073e-01 7.239e-02 4.512e-02 9.493e-13 1.667e-01
 10 11 12 13 14 15 16
1.322e-01 1.848e-01 4.262e-01 1.040e-01 7.986e-02 2.285e-01 1.200e-02
```

```
Eigenvalues for PCoA axes:
(Showing 8 of 2311 eigenvalues)
 PCoA1 PCoA2 PCoA3 PCoA4 PCoA5 PCoA6 PCoA7 PCoA8
733.14 214.18 145.47 48.19 42.45 28.75 26.07 19.35


#now do the same with the "before" subset.

View(converted_test_with_ANN_predicted_15sep2022_orig)

new_df <- converted_test_with_ANN_predicted_15sep2022 %>%
group_by(detailed.label) %>% slice_sample(n=500, replace = TRUE)


 #recode the factor IVs to ordinal numeric data so it can be processed properly. Note that this is
the random-sample raw dataset, e.g. the "before" version showing the curatred labeled DV.
 new_df$proto <- as.factor(new_df$proto)
 new_df$service <- as.factor(new_df$service)
 new_df$conn_state <- as.factor(new_df$conn_state)
 new_df$history <- as.factor(new_df$history)
 new_df$detailed.label <- as.factor(new_df$detailed.label)
 new_df$dl_prediction <- as.factor(new_df$dl_prediction)

 new_df$proto <- as.numeric(new_df$proto)
 new_df$service <- as.numeric(new_df$service)
 new_df$conn_state <- as.numeric(new_df$conn_state)
 new_df$history <- as.numeric(new_df$history)
 new_df$detailed.label <- as.numeric(new_df$detailed.label)


 #get rid of the new DV (from ANN predictive model)
 new_df$dl_prediction <- NULL


 View(new_df)
dis <- vegdist(new_df, na.rm = TRUE)
mod <- betadisper(dis, new_df$detailed.label

        Homogeneity of multivariate dispersions

Call: betadisper(d = dis, group = new_df$detailed.label)

No. of Positive Eigenvalues: 652
No. of Negative Eigenvalues: 1385
```

```
Average distance to median:
 1 2 3 4 5 6 7 8 9
3.652e-01 1.153e-01 9.759e-02 9.281e-02 1.408e-01 1.298e-01 7.369e-02 5.806e-02 7.403e-02
 10 11 12 13 14 15
3.555e-01 1.851e-01 1.869e-01 1.012e-01 2.174e-01 6.436e-05

Eigenvalues for PCoA axes:
(Showing 8 of 2037 eigenvalues)
 PCoA1 PCoA2 PCoA3 PCoA4 PCoA5 PCoA6 PCoA7 PCoA8
695.63 343.28 132.23 58.44 55.14 33.34 21.42 20.34
```

```
#model RQ2 (Shapley cohort values) - 29 SEP 2022

gc()

setwd("C://ISU_PhD//COT711_Dissertation//")
getwd()

library(readr)
converted_test_with_ANN_predicted_15sep2022 <-
read_csv("converted_test_with_ANN_predicted_15sep2022.csv", col_types = cols(..1 =
col_skip(), dl_prediction = col_skip()))


#now convert the character IVs to factor IVs

converted_test_with_ANN_predicted_15sep2022$proto <-
as.factor(converted_test_with_ANN_predicted_15sep2022$proto)
converted_test_with_ANN_predicted_15sep2022$service <-
as.factor(converted_test_with_ANN_predicted_15sep2022$service)
converted_test_with_ANN_predicted_15sep2022$conn_state <-
as.factor(converted_test_with_ANN_predicted_15sep2022$conn_state)
converted_test_with_ANN_predicted_15sep2022$history <-
as.factor(converted_test_with_ANN_predicted_15sep2022$history)
converted_test_with_ANN_predicted_15sep2022$detailed.label <-
as.factor(converted_test_with_ANN_predicted_15sep2022$detailed.label)

library(h2o)
h2o.init(nthreads=-1, max_mem_size = "55G")

final_master_df.h2o <- as.h2o(converted_test_with_ANN_predicted_15sep2022,
key="final_master_df.h2o")

# Split into train & test
```

```
splits <- h2o.splitFrame(final_master_df.h2o, ratios = 0.7, seed = 123)
train <- splits[[1]]
test <- splits[[2]]

#see:

#recover some RAM by removing the original df from R memory. Keep everything in h2o for
now.
converted_test_with_ANN_predicted_15sep2022 <- NULL

gc()

#converted_multiclass_gbm <- h2o.gbm(x = 1:17,
 y = 18,
 model_id = "converted_multiclass_gbm",
 categorical_encoding = "Eigen",
 distribution = "multinomial",
 max_depth = 5,
 ntrees = 100,
 learn_rate = 0.3,
 seed = 12345,
 keep_cross_validation_predictions = FALSE,
 training_frame = train,
 validation_frame = test,
 balance_classes = TRUE)

#revised h2o gbm call, this should run faster (no class balancing, no k-fold cross validation)

converted_multiclass_gbm <- h2o.gbm(x = 1:17,
 y = 18,
 model_id = "converted_multiclass_gbm",
 categorical_encoding = "Eigen",
 distribution = "multinomial",
 max_depth = 5,
 ntrees = 30,
 learn_rate = 1.0,
 seed = 12345,
 keep_cross_validation_predictions = FALSE,
 training_frame = train,
 balance_classes = FALSE)



#now once the GBM is trained, conduct this:
```

```
# Eval performance:
perf <- h2o.performance(converted_multiclass_gbm)

perf


# Generate predictions on a validation set (if necessary):
pred <- h2o.predict(converted_multiclass_gbm, newdata = final_master_df.h2o) #predict on the entire df

# Extract feature interactions:
feature_interactions <- h2o.feature_interaction(converted_multiclass_gbm)


#once all this works and this output is recorded, conduct the Shapley cohort score analysis:

#https://towardsdatascience.com/a-minimal-example-combining-h2os-automl-and-shapley-s-decomposition-in-r-ba4481282c3c

#https://rdrr.io/github/laresbernardo/lares/man/h2o_shap.html

#the below code taken from rdrr.io URL (link above):


#upgrade to h2o 3.38 to get this to work.

# Calculate SHAP values
#SHAP_values <- h2o_shap(model)

#SHAP_values <- h2o_shap(converted_multiclass_gbm) # this doesn't work. see docs
shap_summary_plot <- h2o.shap_summary_plot(converted_multiclass_gbm, test)

#shapley contributions not currently supported for multinomial models in h2o.

print(shap_summary_plot)

#h2o.shap_explain_row_plot() #use this function for Shap explanations of specific rows of data.
#try this on a specific row.
#Calculating contributions is currently not supported for multinomial models.
# DAMN.



expl1 <- h2o.explain(converted_multiclass_gbm, test)
print(expl1)
#doesn't seem to like eigenfactor encoded gbm models. :(
```

```
# Equivalent to:
# SHAP_values <- h2o_shap(
# model = model$model,
# test = model$datasets$test,
# scores = model$scores_test$scores)

# Check SHAP results
head(SHAP_values)

# You must have "ggbeeswarm" library to use this auxiliary function:
# Plot SHAP values (feature importance)
install.packages('ggbeeswarm')
library(ggbeeswarm)

plot(SHAP_values)

# Plot some of the variables (categorical)
shap_var(SHAP_values, Pclass)

# Plot some of the variables (numerical)
shap_var(SHAP_values, Fare)

## End(Not run)


#forget h2o for shapley scores since it can't yet handle multinomial classification shapley scores.
Let's consider using the fastshap package for this, using:
#https://bgreenwell.github.io/fastshap/articles/fastshap.html

#continue here after I skip past the h2o attempts above.
#use a combination ranger forest model and then a fastshap Shapely calculation methodology.
################################################################################
####################

install.packages('fastshap')
library(fastshap)

#create a vector of the IVs (predictor vars), minus the y or DV
X <- subset(converted_test_with_ANN_predicted_15sep2022, select = -detailed.label)


#convert factors to ordinal values.
```

```
converted_test_with_ANN_predicted_15sep2022$proto <-
as.factor(converted_test_with_ANN_predicted_15sep2022$proto)
converted_test_with_ANN_predicted_15sep2022$service <-
as.factor(converted_test_with_ANN_predicted_15sep2022$service)
converted_test_with_ANN_predicted_15sep2022$conn_state <-
as.factor(converted_test_with_ANN_predicted_15sep2022$conn_state)
converted_test_with_ANN_predicted_15sep2022$history <-
as.factor(converted_test_with_ANN_predicted_15sep2022$history)
converted_test_with_ANN_predicted_15sep2022$detailed.label <-
as.factor(converted_test_with_ANN_predicted_15sep2022$detailed.label)

#convert the factor IVs to ordinal IVs
converted_test_with_ANN_predicted_15sep2022$proto <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$proto)
converted_test_with_ANN_predicted_15sep2022$service <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$service)
converted_test_with_ANN_predicted_15sep2022$conn_state <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$conn_state)
converted_test_with_ANN_predicted_15sep2022$history <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$history)

converted_test_with_ANN_predicted_15sep2022$detailed.label <-
as.numeric(converted_test_with_ANN_predicted_15sep2022$detailed.label)

library(ranger)
set.seed(102)
#rfo <- ranger(y ~ ., data = converted_test_with_ANN_predicted_15sep2022)


library(dplyr)

set.seed(123)
#subset this very large dataframe by randomly selecting 75000 rows of data, to be able to fit the
permanova function vector into RAM
#final_master_df_small <- sample_n(final_master_df, 60000)
final_master_df_small <- sample_n(converted_test_with_ANN_predicted_15sep2022, 60000)


y <- final_master_df_small
y$`detailed.label` <- NULL
#y becomes my predictor variables list (IVs), without the DV present.

#based on https://bgreenwell.github.io/fastshap/articles/fastshap.html :

rfo <- ranger(detailed.label ~ ., data = final_master_df_small)
```

```
rfo

Ranger result

Call:
 ranger(detailed.label ~ ., data = final_master_df_small)

Type: Classification
Number of trees: 500
Sample size: 60000
Number of independent variables: 17
Mtry: 4
Target node size: 1
Variable importance mode: none
Splitrule: gini
OOB prediction error: 2.04 %


#2.04% = pretty good ;)

X <- subset(final_master_df_small, select = -detailed.label) # feature columns only



# Prediction wrapper
pfun <- function(object, newdata) {
 predict(object, data = newdata)$predictions
}


# Compute fast (approximate) Shapley values using 10 Monte Carlo repetitions
system.time({ # estimate run time
 set.seed(5038)
 shap <- explain(rfo, X = X, pred_wrapper = pfun, nsim = 10)
})


 mtcars.ppr <- ppr(detailed.label ~ ., data = final_master_df_small, nterms = 1)
Error in ppr.default(X, Y, w, ..) :
 'ppr' applies only to numerical variables

#make the DV an ordinal conversion value
 final_master_df_small$detailed.label <- as.numeric(final_master_df_small$detailed.label)

 # Fit a projection pursuit regression model
 mtcars.ppr <- ppr(detailed.label ~ ., data = final_master_df_small, nterms = 1)
```

```
#model RQ4 (ANN Anomalies and statistical validation vs the curated DV using statistical equivalency) - 15 OCT 2022

gc()

setwd("C://ISU_PhD//COT711_Dissertation//")
getwd()

library(readr)
converted_test_with_ANN_predicted_15sep2022 <-
read_csv("converted_test_with_ANN_predicted_15sep2022.csv", col_types = cols(..1 =
col_skip(), dl_prediction = col_skip()))

library(readr)
 final_master_df <- read_csv("final_master_df.csv", col_types = cols(..1 = col_skip()))

#now from here, I need to convert the curated DV to being either "good" (benign) or bad, so I have a true/false outcome DV to conduct the
#test of statistical equivalence against. I also want to make sure that my benign class data is in the majority.



table(final_master_df$`detailed-label`)

# - 0
# 75955 11139745
# Attack C&C
# 8865 17613
# C&C-FileDownload C&C-HeartBeat
# 53 26962
# C&C-HeartBeat-Attack C&C-HeartBeat-FileDownload
# 834 11
# C&C-Mirai C&C-PartOfAHorizontalPortScan
# 2 888
# C&C-Torii DDoS
# 30 12267750
# FileDownload Okiru
# 18 38317947
# Okiru-Attack Pa
```

```
# 3 1
# PartOfAHorizontalPortScan PartOfAHorizontalPortScan-Attack
# 85067519 5

#need to randomly subsample (reduce) the number of PartOfAHorizontalPortScan observsations
to ~10000 each, along with DDos and Okiru.

#use stratified from Github to do this:

library(devtools) ## To download "stratified"
source_gist("https://gist.github.com/mrdwab/6424112")

subsample1 <- stratified(final_master_df, "detailed-label", 10000)

#now remove the "-" and "0" (benign) classes from this attack subset df, because we want all of
those raw observations, not just 10000.
#remove just the "-" because "0" is malware (zero day attack class)

subsample2 <- subsample1[!(subsample1$`detailed-label` =="-"),]
View(subsample2)

#now do the opposite, KEEP the - and 0 (benign) classes from the original (raw) df.
#subsample3 <- final_master_df[(final_master_df$`detailed-label` =="-" |
final_master_df$`detailed-label`=="0"),]
subsample3 <- final_master_df[(final_master_df$`detailed-label` =="-"),]


#now merge subsample2 and subsample3 into a new final df.
 minority_attack_class_df <- rbind(subsample2, subsample3)

 getwd()
#[1] "C:/ISU_PhD/COT711_Dissertation"
write.csv(minority_attack_class_df, "rq4_minority_attack_class_df15oct2022.csv")


#rename the two benign classes to be the same class so it makes it easier to label the new
benign/malicious DV.
#minority_attack_class_df$`detailed-label` <- gsub('0', '-', minority_attack_class_df$`detailed-
label`)

#now create a new secondary DV which is either "benign" or "attack" to make it easier for
csalculating accuracy

#set default value of new DV to Malicious.
minority_attack_class_df$new_DV <- "Malicious"
View(minority_attack_class_df)
```

```
#if old DV is of class -, then new DV is Benign. That way I don't have to do a whole huge CASE statement for each malware sub-type.
minority_attack_class_df$new_DV[minority_attack_class_df$`detailed-label` == '-'] <- "Benign"
View(minority_attack_class_df)
table(minority_attack_class_df$new_DV)

#fixed.. zero day attacks should now be included here as a part of the malicious DV class.
################

table(minority_attack_class_df$new_DV)

# Benign Malicious
# 75955 70729
#now set up the h2o cluster, and start ingesting the character IVs to be factors (inside the h2o cluster, due to the size of the df).

h2o.init(max_mem_size = "50g", nthreads = -1)
h2o.removeAll()

minority_attack_class_df_hf <- as.h2o(minority_attack_class_df)

#now factor-ize the IVs in the h2o df

minority_attack_class_df_hf["proto"] <- as.factor(minority_attack_class_df_hf["proto"])
minority_attack_class_df_hf["service"] <- as.factor(minority_attack_class_df_hf["service"])
minority_attack_class_df_hf["conn_state"] <- as.factor(minority_attack_class_df_hf["conn_state"])
minority_attack_class_df_hf["history"] <- as.factor(minority_attack_class_df_hf["history"])

#finished performing the IV factorizations.

#now build a deep learning anomaly detecting autoencoding neural network model.

#IV columns 1:17 will be used as predictor input IVs.

#continue with the ANN model now:


#attack_detection_auto_model = h2o.deeplearning(x = 1:17, training_frame = minority_attack_class_df_hf, autoencoder = TRUE, hidden = c(13, 13), adaptive_rate = TRUE, rho = 0.99, epsilon = 1e-08, rate = 0.005, rate_annealing = 1e-06, rate_decay = 1, epochs = 5)

#model parms above resulted in unstable model. simplify (remove the rho, etc. parms and go with defaults).
```

```
#v1
#this works but it is giving weak results (known attack records showing low anomaly sensitivity)
attack_detection_auto_model = h2o.deeplearning(x = 1:17, training_frame =
minority_attack_class_df_hf, autoencoder = TRUE, hidden = c(13, 13), epochs = 5)

malicious_anom = h2o.anomaly(attack_detection_auto_model, minority_attack_class_df_hf)
head(malicious_anom, 20)

malicious_anom_per_feature = h2o.anomaly(attack_detection_auto_model,
minority_attack_class_df_hf, per_feature = TRUE)
head(malicious_anom_per_feature, 20)



#v2
attack_detection_auto_model2 = h2o.deeplearning(x = 1:17, training_frame =
minority_attack_class_df_hf, autoencoder = TRUE, hidden = c(13, 13), epochs = 50)
malicious_anom = h2o.anomaly(attack_detection_auto_model2, minority_attack_class_df_hf)
head(malicious_anom, 20)

malicious_anom_per_feature = h2o.anomaly(attack_detection_auto_model2,
minority_attack_class_df_hf, per_feature = TRUE)
head(malicious_anom_per_feature, 20)


#v3
attack_detection_auto_model3 = h2o.deeplearning(x = 1:17, training_frame =
minority_attack_class_df_hf, categorical_encoding = "Eigen", autoencoder = TRUE, hidden =
c(13, 13), epochs = 50)
malicious_anom = h2o.anomaly(attack_detection_auto_model3, minority_attack_class_df_hf)
head(malicious_anom, 20)

malicious_anom_per_feature = h2o.anomaly(attack_detection_auto_model3,
minority_attack_class_df_hf, per_feature = TRUE)
head(malicious_anom_per_feature, 20)


#v4, columns 1:16 only (dropping tunnel_parents). reducing to 1 hidden layer to see if it helps
stability.
attack_detection_auto_model4 = h2o.deeplearning(x = 1:16, training_frame =
minority_attack_class_df_hf, categorical_encoding = "Eigen", autoencoder = TRUE, hidden =
c(10), epochs = 50)


#much better with v5 below.
```

```
#v5, columns 1:16 only (dropping tunnel_parents). reducing to 1 hidden layer to see if it helps stability. used the tanh activation function to prevent exponential growth.
attack_detection_auto_model5 = h2o.deeplearning(x = 1:16, training_frame = minority_attack_class_df_hf, categorical_encoding = "Eigen", activation = c("Tanh"), autoencoder = TRUE, hidden = c(10), epochs = 50)


malicious_anom = h2o.anomaly(attack_detection_auto_model5, minority_attack_class_df_hf)
head(malicious_anom, 20)

malicious_anom_per_feature = h2o.anomaly(attack_detection_auto_model5, minority_attack_class_df_hf, per_feature = TRUE)
head(malicious_anom_per_feature, 20)

#nice on v5 features and anomaly scores above, we are actually detecting things now LOL :)


#v5, columns 1:16 only (dropping tunnel_parents). reducing to 1 hidden layer to see if it helps stability. used the tanh activation function to prevent exponential growth.
attack_detection_auto_model6 = h2o.deeplearning(x = 1:16, training_frame = minority_attack_class_df_hf, categorical_encoding = "Eigen", activation = c("Tanh"), autoencoder = TRUE, hidden = c(10), epochs = 50)


malicious_anom = h2o.anomaly(attack_detection_auto_model5, minority_attack_class_df_hf)
head(malicious_anom, 20)

malicious_anom_per_feature = h2o.anomaly(attack_detection_auto_model5, minority_attack_class_df_hf, per_feature = TRUE)
head(malicious_anom_per_feature, 20)


#v6, improved v5. columns 1:16 only (dropping tunnel_parents). now 2 hidden layers. used the tanh activation function to prevent exponential growth.
attack_detection_auto_model6 = h2o.deeplearning(x = 1:16, training_frame = minority_attack_class_df_hf, categorical_encoding = "Eigen", activation = c("Tanh"), autoencoder = TRUE, hidden = c(11, 11), epochs = 50)


malicious_anom = h2o.anomaly(attack_detection_auto_model6, minority_attack_class_df_hf)
head(malicious_anom, 20)

malicious_anom_per_feature = h2o.anomaly(attack_detection_auto_model6, minority_attack_class_df_hf, per_feature = TRUE)
head(malicious_anom_per_feature)
```

#great job with v6! Now write the anomaly score output from v6 into the origiinal minority_attack_class df.

```
anomaly_score <- as.data.frame(malicious_anom)
minority_attack_class_df$anomaly_score <- anomaly_score$Reconstruction.MSE
```

#works great.

*** use this scatterplot in the Discussion section for RQ4.
#create a scatterplot of univariate data (anomaly_score DV). Note that it will be really slow because (gulp..) there are ~11.2m rows of data to get plotted..
#this takes ~25 mins to be produced on my machine.

```
library(ggplot2)
ggplot(minority_attack_class_df, aes(x = 1:nrow(minority_attack_class_df), y =
anomaly_score)) + geom_point()
```

#But what really want to do is create a color-coded scatterplot of the anomaly_scores by the two groups, e.g. benign vs malicious.

```
ggplot(minority_attack_class_df, aes(x = 1:nrow(minority_attack_class_df), y = anomaly_score,
color=newDV)) + geom_point()
```

#boxplots of the two groups

```
ggplot(minority_attack_class_df, aes(x = 1:nrow(minority_attack_class_df), y = anomaly_score,
color = new_DV)) + geom_boxplot()
```

#also per rq4, want to calculate the anomaly score means between those same 2 groups.
#get mean(anomaly_score) for each of the two groups in new_DV

#then lastly conduct a permanova for variance between two groups.

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

Benign Malicious
75955 70729

#subset my data into two groups

```
benign <- minority_attack_class_df %>% filter(str_detect(new_DV,"Benign"))
malicious <- minority_attack_class_df %>% filter(str_detect(new_DV,"Malicious"))
```

#now create a numeric-converted version of the factors in the df.

```
minority_attack_class_df_numeric$proto <- as.factor(minority_attack_class_df_numeric$proto)
minority_attack_class_df_numeric$proto <-
as.numeric(minority_attack_class_df_numeric$proto)

minority_attack_class_df_numeric$service <-
as.factor(minority_attack_class_df_numeric$service)
minority_attack_class_df_numeric$service <-
as.numeric(minority_attack_class_df_numeric$service)

minority_attack_class_df_numeric$conn_state <-
as.factor(minority_attack_class_df_numeric$conn_state)
minority_attack_class_df_numeric$conn_state <-
as.numeric(minority_attack_class_df_numeric$conn_state)

minority_attack_class_df_numeric$history <-
as.factor(minority_attack_class_df_numeric$history)
minority_attack_class_df_numeric$history <-
as.numeric(minority_attack_class_df_numeric$history)


#now conduct a beta dispersion test between the two groups.
library(vegan)
#df[c(1, 2, 4)]
#dis <- vegdist(varespec)
#dis <- vegdist(minority_attack_class_df_numeric[c(1:16,20)]


# data <- (minority_attack_class_df_numeric2[c(1:16,20)])
data <- (minority_attack_class_df_numeric2[c(1:8,11:16,18,19)])

View(data)
nrow(data)
#[1] 146667


data2 <- sample_n(data, 40000)
#40000 observations of 16 variables.

DVgroup <- data2$newDV

data2$new_DV <- NULL #remove the factor DV from data2.

gc()

#Description
```

#Implements Marti Anderson's PERMDISP2 procedure for the analysis of multivariate homogeneity of group dispersions (variances).
betadisper is a multivariate analogue of Levene's test for homogeneity of variances. Non-euclidean distances between objects and group centres (centroids or medians) are handled
by reducing the original distances to principal coordinates. This procedure has latterly been used as a means of assessing beta diversity. There are anova, scores, plot and boxplot
methods.

#TukeyHSD.betadisper creates a set of confidence intervals on the differences between the mean distance-to-centroid of the levels of the grouping factor with the specified family-wise probability of coverage. The intervals are based on the Studentized range statistic, Tukey's 'Honest Significant Difference' method.

#calculate the Bray-Curtis distances between samples. using 40k randomly selected observations from the original balanced training dataset of 146,684 observations.
dis <- vegdist(data2, na.rm = TRUE)

mod <- betadisper(dis, DVgroup) # in progress on 5:39pm on Sun., 10/16.

#####continue from here on Sunday evening. Capture all of these outputs for the RQ4 results section of the dissertation:

mod

#Homogeneity of multivariate dispersions
#
#Call: betadisper(d = dis, group = DVgroup)
#
#No. of Positive Eigenvalues: 1887
#No. of Negative Eigenvalues: 3560
#
#Average distance to median:
Benign Malicious
0.3640 0.2801
#
#Eigenvalues for PCoA axes:
#(Showing 8 of 5447 eigenvalues)
PCoA1 PCoA2 PCoA3 PCoA4 PCoA5 PCoA6 PCoA7 PCoA8
#2762.98 1920.99 1436.38 507.96 273.90 159.07 113.66 73.54

Perform test
anova(mod)

```
## Permutation test for F
permutest(mod, pairwise = TRUE, permutations = 99)

#Permutation test for homogeneity of multivariate dispersions
#Permutation: free
#Number of permutations: 99
#
#Response: Distances
# Df Sum Sq Mean Sq F N.Perm Pr(>F)
#Groups 1 70.3 70.297 1160.5 99 0.01 **
#Residuals 39998 2422.9 0.061
#---
#Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
#Pairwise comparisons:
#(Observed p-value below diagonal, permuted p-value above diagonal)
# Benign Malicious
#Benign 0.01
#Malicious 9.1268e-251

## Tukey's Honest Significant Differences
(mod.HSD <- TukeyHSD(mod))
plot(mod.HSD)

## Plot the groups and distances to centroids on the
## first two PCoA axes
plot(mod)

#continue here when I get home the weekend of 10/21.
## with data ellipses instead of hulls
plot(mod, ellipse = TRUE, hull = FALSE) # 1 sd data ellipse
plot(mod, ellipse = TRUE, hull = FALSE, conf = 0.90) # 90% data ellipse

plot(mod, main = "IoT-23 Multivariate Homogeneity of Group Dispersions", ellipse = TRUE,
hull = FALSE, conf = 0.90) # 90% data ellipse

scores(mod)

#then we are DONE. :)

################################################################################
##############################################


mod <- betadisper(dis, DVgroup)
```

```
Error in readChar(con, 5L, useBytes = TRUE) : cannot open the connection
In addition: Warning message:
In readChar(con, 5L, useBytes = TRUE) :
 cannot open compressed file 'C:/Users/twool/AppData/Local/Temp/RtmpstsYPG/rs-graphics-
3d781a85-a4bb-44ea-98aa-fcc68eb32024/f9658e1f-46d3-4ed0-acb2-870243e54edf.snapshot',
probable reason 'No such file or directory'
Graphics error: Plot rendering error
> mod

        Homogeneity of multivariate dispersions

Call: betadisper(d = dis, group = DVgroup)

No. of Positive Eigenvalues: 1887
No. of Negative Eigenvalues: 3560

Average distance to median:
 Benign Malicious
 0.3640 0.2801

Eigenvalues for PCoA axes:
(Showing 8 of 5447 eigenvalues)
 PCoA1 PCoA2 PCoA3 PCoA4 PCoA5 PCoA6 PCoA7 PCoA8
2762.98 1920.99 1436.38 507.96 273.90 159.07 113.66 73.54
>
> ## Perform test
> anova(mod)
Analysis of Variance Table

Response: Distances
 Df Sum Sq Mean Sq F value Pr(>F)
Groups 1 70.3 70.297 1160.5 < 2.2e-16 ***
Residuals 39998 2422.9 0.061
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> ## Permutation test for F
> permutest(mod, pairwise = TRUE, permutations = 99)

Permutation test for homogeneity of multivariate dispersions
Permutation: free
Number of permutations: 99

Response: Distances
 Df Sum Sq Mean Sq F N.Perm Pr(>F)
Groups 1 70.3 70.297 1160.5 99 0.01 **
```

```
Residuals 39998 2422.9 0.061
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Pairwise comparisons:
(Observed p-value below diagonal, permuted p-value above diagonal)
 Benign Malicious
Benign 0.01
Malicious 9.1268e-251
>
> ## Tukey's Honest Significant Differences
> (mod.HSD <- TukeyHSD(mod))
 Tukey multiple comparisons of means
 95% family-wise confidence level

Fit: aov(formula = distances ~ group, data = df)

$group
 diff lwr upr p adj
Malicious-Benign -0.08388974 -0.08871624 -0.07906324 0

> plot(mod.HSD)
>
> ## Plot the groups and distances to centroids on the
> ## first two PCoA axes
> plot(mod)
>
```